SEP SES TNM

INSTITUTO TECNOLÓGICO DE CHIHUAHUA II

SOFTWARE PARA LA ESTIMULACIÓN VISUAL DE NIÑOS CON BAJA VISIÓN

TESIS
PARA OBTENER EL GRADO DE

MAESTRA EN SISTEMAS COMPUTACIONALES

PRESENTA

ING. ROSALBA LÓPEZ MENDOZA

DIRECTOR DE TESIS
M.C. LEONARDO NEVÁREZ CHÁVEZ

CO-DIRECTOR DE TESIS
DRA. MARISELA IVETTE CALDERA FRANCO

CHIHUAHUA, CHIH; JUNIO DEL 2019.

Dictamen

Chihuahua, Chih., 29 de mayo del 2019

LIC. OLGA REBECA CASTILLO CRUZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
Presente.-

Por medio de este conducto el comité tutorial revisor de la tesis para obtención de grado de Maestro en Sistemas Computacionales, que lleva por nombre **"SOFTWARE PARA LA ESTIMULACIÓN VISUAL DE NIÑOS CON BAJA VISIÓN"**, que presenta el (la) **C. ROSALBA LÓPEZ MENDÓZA**, hace de su conocimiento que después de ser revisado ha dictaminado la **APROBACIÓN** del mismo.

Sin otro particular de momento, queda de Usted.

Atentamente

La Comisión de Revisión de Tesis.

Leonardo Nevárez Chávez

M.C. LEONARDO NEVÁREZ CHÁVEZ
Director de Tesis

M.E.S. MARISELA IVETTE CALDERA FRANCO
Co-Director

DR. ALBERTO CAMACHO RIOS
Revisor

M.C. ARTURO LEGARDA SÁENZ
Revisor

SECRETARIA DE EDUCACION PUBLICA
INSTITUTO TECNOLOGICO DE CHIHUAHUA II
DIVISION DE ESTUDIOS DE POSGRADO E INVESTIGACION

AGRADECIMIENTOS

Agradezco la oportunidad de poder haber realizado este proyecto de investigación, y de haber aprendido nuevos conocimientos en el trayecto. Agradezco también, el apoyo y la paciencia de mis maestros quienes fueron guiándome en el transcurso de su realización. En especial quiero agradecer a mi director de tesis el M.C. Leonardo Nevárez Chávez, al Dr. Alberto Camacho Ríos, a la Dra. Marisela Ivette Caldera Franco y demás profesores que fueron parte de mi formación académica dentro del programa de posgrado. También quiero agradecer al Centro de Estudios para Invidentes A.C. por la confianza brindada para realizar este proyecto, al Ing. José Luis Tam, al Ing. Fernando Beltrán y a la Lic. Selene Armendáriz. Por último, quiero agradecer al Consejo Nacional de Ciencia y Tecnología CONACYT por el apoyo económico brindado para la realización del proyecto.

RESUMEN

La actualidad refiere una era donde el uso de las Tecnologías de la Información es cada vez más común en diferentes áreas. La inclusión de estos medios en el proceso de enseñanza–aprendizaje les convierte en valiosos materiales de estudio. La discapacidad visual en los niños tiene un alto impacto en su educación ya que su aprendizaje puede verse retrasado e impedir adquirir las habilidades necesarias para llevar una vida plena. Contar con un software diseñado especialmente para el aprendizaje y la estimulación visual de niños con deficiencias visuales, puede servir de gran apoyo en su rehabilitación y de esta manera propiciar una más rápida adaptación a la vida diaria. Por lo cual se desarrolló una aplicación informática que sirva de apoyo para dar terapia de estimulación visual a los niños con baja visión dando como resultado una útil y valiosa herramienta, que ayudará mucho en las terapias que tomen los niños que asisten al Centro de Estudios para Invidentes CEIAC.

ABSTRACT

The current situation refers to an era where the use of Information Technology is increasingly common in different areas. The inclusion of these means in the teaching-learning process makes them valuable study materials. Visual impairment in children has a major impact on their education because their learning may be delayed and prevent them from acquiring the skills needed to lead a full life. Having software specially designed for the learning and visual stimulation of visually impaired children, can be a great support in their rehabilitation and, in this way, promote a faster adaptation to daily life. For this reason, a computer application was developed to support visual stimulation therapy for children with low vision, resulting in a useful and valuable tool that will help a lot in the therapies for children attending in the Centro de estudios para invidentes CEIAC.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

1.1. Introducción

Según la Organización Mundial de la Salud (OMS) en el planeta hay aproximadamente 285 millones de personas con discapacidad visual, de las cuales 39 millones son determinantemente ciegas y 246 millones presentan baja visión. Aproximadamente un 90% de la carga mundial de discapacidad visual se concentra en los países de ingresos bajos. El 82% de las personas que padecen ceguera tienen 50 años o más (OMS, 2014) (ver figura 1.1).

La misma OMS, estima que el número de niños con discapacidad visual asciende a 19 millones, de los cuales 12 millones la padecen debido a errores de refracción, fácilmente diagnosticables y corregibles. Unos 1.4 millones de menores de 15 años sufren ceguera irreversible y necesitan intervenciones de rehabilitación visual para su pleno desarrollo psicológico y personal (OMS, 2014).

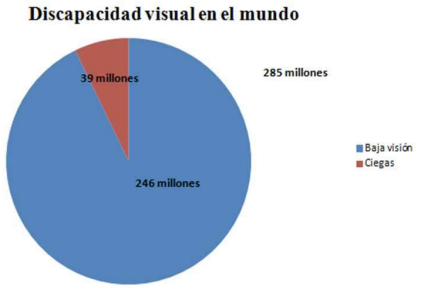

Figura 1.1 Estadística de discapacidad en el mundo

INTRODUCCIÓN

Para un niño con visión normal, pasar por las etapas evolutivas que se producen mientras aprende a ver es relativamente fácil, pero un déficit visual grave inhibe este proceso, ya que el desarrollo de las estructuras visuales, de la función de la retina, de las vías ópticas y/o del área cortical, se ve gravemente comprometido, lo que repercute en un limitado bagaje de experiencias visuales por la incapacidad para recoger información incidentalmente a través de la vista. (Pérez, 2015).

Por lo tanto, para un niño con baja visión las oportunidades de aprendizaje que se le proporcionen son fundamentales, y requieren un conjunto de acciones específicas dirigidas, por ejemplo:

- Debe recibir una estimulación visual lo más temprana posible que potencie sus experiencias visuales y su maduración general.
- En su entorno familiar y social, se ha de entender y aceptar la naturaleza de sus dificultades, cuyas respuestas visuales, aunque no son ciegos, no son las que manifiestan los niños con visión normal.

Al mismo tiempo, los niños con baja visión presentan una serie de características específicas y permanentes que siempre se deben tomar en cuenta (Pérez, 2015), estas son:

- Las dificultades visuales reducen el aspecto globalizador de la visión. La percepción de los objetos se produce de una manera analítica, lo que produce un ritmo más lento del aprendizaje.
- Dificultades para imitar conductas, gestos y juegos observados visualmente, por lo que siempre necesitará una atención personalizada que le ayude a entender lo que pasa a su alrededor, para ser capaz de asimilarlo y reproducirlo.
- Su autoimagen puede estar alterada como consecuencia de las frustraciones que recibe al darse cuenta de que no reacciona como los demás. Se pondrá especial atención tanto en reforzar esta autoimagen como en mitigar los miedos que, con frecuencia, presenta ante situaciones aparentemente normales para sus iguales.

- Mayor fatiga a la hora de realizar sus actividades por el mayor esfuerzo que debe hacer ante cualquier tarea visual.

Como consecuencia de sus dificultades para la visión de lejos, hay una menor información del medio que le rodea, tanto en calidad como en cantidad (Pérez, 2015).

Por estimulación visual entendemos la serie ordenada de experiencias visuales, según la edad y maduración del niño, encaminadas a que su desarrollo visual se aproxime al considerado como normal (Pérez, 2015).

La estimulación visual, tiene como objetivo mejorar el funcionamiento de los niños con baja visión, es decir, aquellos que tienen una reducción de su agudeza o una pérdida del campo visual, debido a una patología ocular o cerebral congénita o adquirida, y que ni siquiera con correcciones ópticas pueden llegar a alcanzar una visión normalizada (García, 2012).

1.2. Planteamiento del problema

El Centro de Estudios para Invidentes, A.C. (CEIAC); es una institución de apoyo para personas ciegas y con baja visión que nace en la ciudad de Chihuahua, Chih.; durante el año de 1995 como asociación civil, para intentar resolver la problemática que enfrentan las personas con algún impedimento visual. Inicia atendiendo a dos jóvenes ciegas que no habían concluido la educación primaria y enfrentaban el reto de continuar sus estudios en el sistema regular (CEIAC, 2017).

En el CEIAC se brinda a las personas ciegas y débiles visuales las habilidades y el apoyo para desarrollar al máximo su potencial humano, rompiendo con los esquemas tradicionales de la educación especial y buscando su integración plena a la realidad de aquellos que ven sin esa dificultad.

En este centro (CEIAC) se tiene la necesidad de contar con un software que sirva de apoyo para dar terapia de estimulación visual a los niños de hasta 7 años de edad. Actualmente se utiliza un

software que lleva por nombre senswitcher, el cual está obsoleto y desactualizado lo que genera un desinterés por parte del paciente a la hora de tomar la terapia de estimulación visual.

El proyecto de software que se va a desarrollar para el CEIAC ofrecerá poder trabajar con una variedad de animaciones, figuras, imágenes y diferentes colores, los cuales podrán cambiarse y actualizarse, también se podrá trabajar con diferentes tipos de ejercicios que propicien una mejora en su agudeza visual, además de que el software contará con un control que permita aumentar o disminuir la velocidad con la que se despliegan las animaciones en la aplicación, esto es con el fin de poder personalizar los ejercicios que va a tomar el niño según la necesidad o el padecimiento con que cuente.

1.3. Alcances y limitaciones

Dentro de los alcances que presenta el software a desarrollar se tiene que:

- El software se podrá utilizar con niños con baja visión.
- El software se podrá utilizar con equipos de escritorio.
- Se puede ejecutar en diferentes sistemas operativos:
  Por ejemplo: Windows, Mac OS.

Por otro lado, tenemos las siguientes limitaciones que tiene el software:

- El software no se podrá usar en dispositivos móviles.
- El software no podrá ser utilizado en niños invidentes (ceguera total).

1.4. Justificación

La mayor parte de los programas utilizados en las escuelas, sobre todo las públicas, son elaborados para ser utilizados por personas sin discapacidad. De esto último nace la necesidad de contar con un software que sea desarrollado para ser utilizado por niños con discapacidades visuales que les permita adaptarse más rápido a la vida diaria y no retrasar su aprendizaje, tener un software actualizado y adaptado a necesidades reales será mejor y más eficaz en el aprendizaje y estimulación del niño.

1.5 Objetivo

Desarrollar un software que sirva de apoyo a los niños con baja visión para mejorar su agudeza visual.

1.6 Objetivos especificos

El programa deberá ser atractivo y dinámico para los niños utilizando diferentes figuras, imágenes, sonidos y colores. Debe además servir como una herramienta de apoyo a los terapeutas del CEIAC quienes utilizarán la aplicación con los niños de baja visión.

II. ESTADO DEL ARTE

En este capítulo se describen los temas que están relacionados con el proyecto, se describen los programas informáticos que han sido utilizados para la estimulación visual de niños con problemas de baja visión.

El punto de partida de cualquier programa de mejora de la eficiencia visual es considerar el tipo de personas al que va dirigido (Barraga et al, 1977; Tobin y Chapman, 1977:1986).

En el contexto de la multiplicidad de variables que deben tomar en cuenta para hacer adecuadas las intervenciones a quienes van dirigidas, el ordenador personal permite una gran flexibilidad en las mismas y, por ello, constituye una herramienta de enorme potencial debido a que, entre otras cosas, integra símbolos verbales y gráficos, propiciando el desarrollo de capacidades cognitivas relacionadas con la atención, la percepción, la discriminación, la categorización, la asociación y la memoria. (Cantón, 1990; De la orden, 1990). Permite mejorar la dinámica del proceso de aprendizaje mediante el auto entrenamiento, dando oportunidad a que se dé un alto grado de individualización ("enseñanza adaptada") al tiempo que potencia que el "aprendizaje" tenga significado para el "aprendiz" (Cantón, 1990). Incorpora presentaciones multimedia en plataformas de trabajo interactivas (De la orden, 1990), lo cual permite adaptar la información en base a la acción directa de los usuarios, así como a sus perfiles de comportamiento.

Los ámbitos en los que se ha hecho uso de los ordenadores en el contexto de la estimulación visual son tres: la fotoestimulación, la medición optométrica y el entrenamiento de capacidades visuales específicas mediante juegos (Rodríguez Soler1, 1997).

Fotoestimulación: Los monitores de los ordenadores son una fuente de luz fácilmente adaptable a las condiciones ambientales y a diferentes tipos de usuarios (Goodrich, 1984; Karshmer et al, 1994). Además, el hecho de que las imágenes en pantalla se crean activando secuencialmente sus distintas partes (lo que usualmente se denomina "barrido de la imagen"), puede utilizarse como fuente de estimulación rítmica para los ojos.

Medición optométrica: La evaluación de la agudeza visual y de la sensibilidad al contraste se realizan cada vez con más frecuencia mediante tecnologías informáticas (Spencer y Ross, 1989; Navarro et al, 1990; Hammerlund, 1994; Arditi y Lei Liu, 1996). Por otra parte, la propia naturaleza de la estimulación visual proporcionada por los monitores, su carácter dinámico y cambiante, ha fomentado que la optometría moderna los incorpore para obtener predictores visuales específicos a la rapidez perceptiva, la agudeza visual dinámica, etc. (Groffman y Solan, 1994). La obtención de este tipo de medidas sólo es posible cuando el soporte informático permite controlar adecuadamente las características estimulares (tamaño e intensidad) de lo presentado en pantalla. En este respecto, tiene especial importancia que la cantidad de luz enviada por cada parte de esta pantalla, es decir su luminancia (Lillo, 2000), no sea otra que la deseada.

Para concluir, se debe resaltar el hecho de que en este ámbito cobra especial relevancia la posibilidad ya mencionada, de registrar las respuestas dadas por los usuarios y su utilización para el empleo de algoritmos de corrección y adaptación (la prueba cambiaría en función de las respuestas). Este hecho les permite convertirse en herramientas de gran robustez métrica (véase, por ejemplo, Berla, 1980; Rodríguez Soler, 1998).

Juegos y entrenamiento de capacidades específicas: La literatura científica cuenta a “Lilli y Gogo” entre los primeros juegos de ordenador empleados como herramientas de estimulación visual (Meralla y Jaritz, 1994). Con la misma finalidad fueron creados una serie de juegos desarrollados en el Tomteboda Resource Centre (The Truck, Worm Max, Look Here, etc.) cuya descripción puede encontrarse en (Hammarlund 1994 y Tobin et al, 1996). En los dos casos citados se emplearon materiales estimulares relativamente simples: contrastes altos, colores y formas geométricas básicas, sonido y movimiento. En ambos casos se constató el alto poder de atracción, que este tipo de materiales ejerce sobre los niños con deficiencia visual, así como la posibilidad de empleo del mismo para fomentar el desarrollo de capacidades visuales específicas.

A continuación, se mencionan algunas aplicaciones que están relacionadas al software que se pretende desarrollar:

Aplicación Senswitcher

Es un programa dirigido a personas con graves dificultades para el aprendizaje. Está compuesto de 132 actividades distribuidas en ocho fases, que abarcan desde aspectos de estimulación visual y auditiva, aprendizaje de la relación causa-efecto y la anticipación. Se puede adecuar a las necesidades de cada alumno, ya que cada fase permite ser configurada de forma sencilla con distintas opciones de pantalla: colores de fondo y objetos/imágenes animadas, mayor lentitud o rapidez en su presentación. Puede ser utilizado mediante teclado, ratón o pulsador (Escritorio modalidad educación especial, s.f.) (Ver figura 2.1).

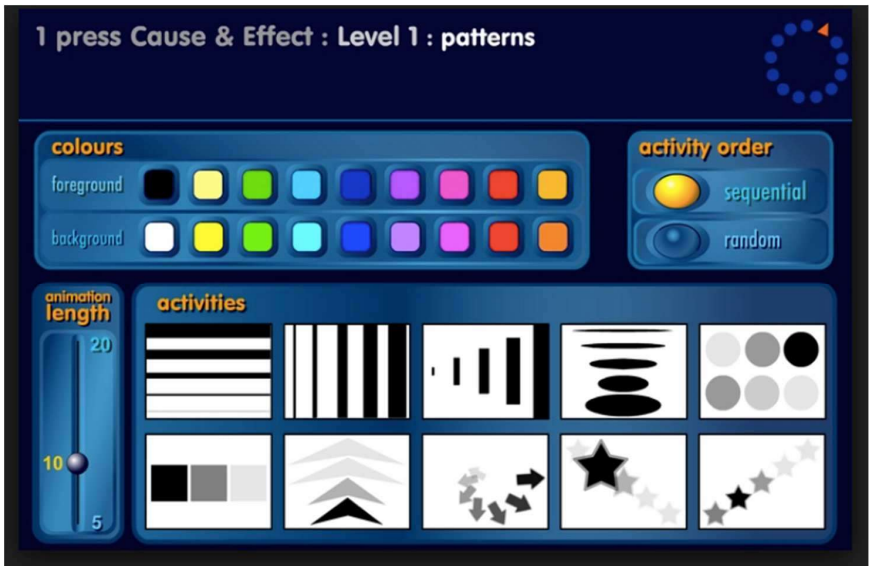

Figura 2.1 Aplicación Senswitcher.

Toque Mágico

Este programa tiene por objeto el desarrollo de nociones y conceptos previos a la escolarización, tales como el lenguaje, los números, la orientación espacial y temporal; de manera entretenida para el niño. El Toque Mágico es ideal para el aprendizaje de niños con dificultades visuales, de entre tres y seis años de edad, que tienen su primer contacto con los días de la semana, los números, cuentos, palabras o conceptos como izquierda o derecha, etc. (Centro de Desarrollo de Tecnologías de Inclusión, 2016) (ver figura 2.2).

Figura 2.2 Aplicación Toque Mágico.

EVO: Sistema informático de entrenamiento visual para personas deficientes visuales

El sistema informático EVO, en el ámbito de la estimulación visual, permite la evaluación de capacidades visuales y facilita la aplicación de programas de entrenamiento visual, basados en juegos de ordenador adaptados a las características visuales del usuario.

Las investigaciones previas únicamente abordaron algunas áreas visuales básicas, tal y como pueden ser la fijación, la localización, el trazado y la exploración. El programa EVO, además de incorporarlas se ocupa de otras áreas visuales, como son la discriminación visual, la categorización, el reconocimiento de formas por siluetas, contornos y rasgos críticos, la reconstrucción a través de modelos incompletos, el reconocimiento de formas simétricas, el dibujo, etc.

El programa EVO permite una gran flexibilidad y amplitud en el empleo de las características estimulares. Esto es, además de permitir una gran versatilidad para controlar los parámetros estimulares normalmente contemplados por otros programas (tamaños, contrastes, colores, velocidades, etc.; véase Tobin et al, 1996), incorpora el empleo de una amplia gama de materiales (formas geométricas, dibujos, fotografías, etc.)

Además, EVO contempla la posibilidad de efectuar, en el marco del propio programa, evaluaciones periódicas de la funcionalidad visual de los usuarios. Gracias a ello la misma plataforma puede ser utilizada por los profesionales de la rehabilitación. Más importante aún, EVO se diseñó para permitir que, como recomiendan Bozic y Tobin (1993), los resultados de tales evaluaciones se puedan utilizar para ajustar las características visuales de los distintos subprogramas de entrenamiento visual incluidos en él (Asociación Doce, 2016).

Cantaletras

Cantaletras es un software pensado para el aprendizaje deximena la escritura en niños ciegos y/o de baja visión, de entre 4 y 10 años. Consiste en una herramienta diseñada no solo desde el punto de vista del aprendizaje, sino también del educador, que tiene a su disposición recursos y consejos para reforzar el modelo de enseñanza. El software ha sido renovado recientemente e incorpora una interfaz amigable que resulta inclusiva y atractiva para todos los niños, sean o no ciegos (Centro de Desarrollo de Tecnologías de Inclusión, 2016) (ver figura 2.3).

.

Figura 2.3 Aplicación Cantaletras.

III. MARCO TEÓRICO

En este capítulo se describen los temas que fueron utilizados para el desarrollo de la tesis.

3.1 Modelo Montessori

Hace alrededor de cien años la Dra. Montessori (1870 – 1952) concibió un nuevo método educativo basado en la estimulación y el respeto. Al complementar este método con su formación en medicina, psicología y antropología, desarrolló su filosofía de la educación basándose en observaciones reales a los niños. Este sistema de educación es a la vez una filosofía de desarrollo del niño y un fundamento para orientar ese crecimiento. Se fundamenta en dos importantes necesidades del desarrollo de los más pequeños:

1. La necesidad de la libertad dentro de los límites.
2. Un entorno cuidadosamente preparado que garantiza la exposición a los materiales y experiencias.

El objetivo principal de un programa Montessori es ayudar a que cada niño alcance su máximo potencial en todos los ámbitos de la vida. Las actividades promueven el desarrollo de habilidades sociales, el crecimiento emocional y la coordinación física, así como la preparación cognitiva para los futuros esfuerzos académicos intelectuales.

El método Montessori permite que el niño experimente la alegría de aprender, el tiempo para disfrutar el proceso y asegure el desarrollo de su autoestima. Proporciona las experiencias a través de las cuales los niños crean sus conocimientos y les prepara para las varias experiencias que ofrece la vida.

La Dra. Montessori descubrió que los niños pasan por fases de interés y curiosidad, que ella denominaba “periodos sensibles” del desarrollo en esta etapa temprana de su vida. Ella misma, describe la mente del niño desde el momento del nacimiento hasta los seis años de edad como

la “mente absorbente “. Es durante esta etapa que un niño tiene una tremenda capacidad de aprender y asimilar el mundo que le rodea, sin esfuerzo consciente. Durante este tiempo, los niños son particularmente receptivos a ciertos estímulos externos. Un(a) “guía” Montessori reconoce y se aprovecha de estas etapas muy perceptivas, a través de la introducción de los materiales y las actividades que están especialmente diseñadas para estimularle.

Es importante mencionar que este enfoque puede servir para ser utilizado en las terapias de estimulación visual ya que son técnicas que el terapeuta puede emplear con el niño al momento de utilizar los ejercicios, por lo que siempre se busca que el niño se sienta motivado a la hora de estar tomando la terapia y que pueda aprovechar al máximo los ejercicios y la terapia que se le está brindando (Familias en ruta, s.f.).

3.1.1 El ambiente preparado

Para que el aprendizaje auto dirigido tenga lugar, todo el ambiente de aprendizaje (aula, materiales y entorno social) debe ser de apoyo para el niño. El guía/facilitador proporciona los recursos necesarios, incluidas las oportunidades de los niños para funcionar en un ambiente seguro y positivo. En conjunto, el “guía” y el niño forman una relación basada en la confianza y el respeto que fomenta la autoconfianza y la voluntad de probar cosas nuevas.

Todo el salón Montessori está diseñado para permitir que el niño llegue a ser independiente, los materiales son de tamaño infantil y el equipo se presenta de una manera ordenada en estantes bajos que son de fácil acceso para ellos. El equipo es estéticamente agradable y se cuida meticulosamente para animar a los niños a cuidar de los materiales también. Los niños entre las edades de dos y medio y seis años, se agrupan en su propia “mini“sociedad. Los niños más pequeños aprenden viendo a los niños mayores y los mayores se benefician al ayudar a los niños más pequeños. El grupo de edad mixto permite a los niños desarrollarse social, intelectual y emocionalmente (Familias en ruta, s.f.).

3.2 Colores y figuras que se utilizan en el software para la estimulación visual

Los niños de baja visión muestran una gran variabilidad en relación no sólo con el resto visual que puedan presentar sino con la funcionalidad que han desarrollado. Entre los niños considerados con baja visión se hallan los que presentan percepción de luz (PL), los que pueden verlos movimientos de la mano (MM), contar dedos (CD) o presentan una agudeza visual inferior a 5/100.

Asimismo, hemos observado que un gran número de estos niños muestran una funcionalidad sorprendente en relación con el escaso resto visual que poseen y que no puede justificarse por la agudeza o por el campo visual que presentan. Estos niños pueden realizar variadas respuestas frente a la presencia de un estímulo. Para ello utilizan guías o puntos de referencia visuales, auditivos, memoria, orientación en el espacio, movilidad y, en conjunto, aprovechan todos los detalles presentes en las figuras o los objetos que les ayudan en su trabajo de identificación. Numerosos bebés usan también su visión periférica durante largo tiempo antes de poder realizar (si les es posible) una fijación más o menos macular.

Algunos ejemplos de figuras utilizadas que sirvan de apoyo en la estimulación visual se pueden encontrar:

- círculos concéntricos
- círculos con líneas convergentes
- diámetros
- contornos con líneas diagonales
- puntos gruesos

3.3 ¿Qué es la deficiencia visual?

La ONCE (Organización Nacional de Ciegos Españoles) establece que una persona con deficiencia visual es aquella persona que "con la mejor corrección posible solamente puede ver o distinguir, aunque con gran dificultad, algunos objetos a una distancia muy corta". Por tanto, la ONCE determina que, las personas con deficiencia visual, a diferencia de aquellas con ceguera, conservan todavía un resto de visión útil para su vida diaria (desplazamiento, tareas domésticas, lectura, etc.). En términos ópticos, de la misma manera que la ceguera, el grado de deficiencia visual se cuantifica mediante la agudeza visual del paciente y su campo visual. Dicha deficiencia visual se clasifica en moderada (la persona percibe objetos y caracteres impresos a pocos centímetros sin necesidad de ayudas ópticas) y severa (la persona percibe volúmenes y colores, objetos y caracteres impresos a pocos centímetros con ayudas ópticas, gafas, lupas, etc.) (Pérez, 2015).

3.4 Discapacidad visual

El concepto de discapacidad visual legalmente incluye los términos de ceguera y deficiencia visual de toda persona cuya visión en ambos ojos presenta al menos una de las siguientes condiciones: baja agudeza visual (AG), campo visual (CV) disminuido e incapacidad para distinguir la intensidad luminosa (Pérez, 2015).

3.4.1 Características específicas de la discapacidad visual

A continuación se describen específicamente las características de la discapacidad visual:

Baja Visión (Débil visual)

Realidad muy heterogénea en la que pesan los siguientes factores:

- Agudeza visual deformada.
- La distinción de colores y/o contraste.
- Presencia de nistagmos (temblor neurológico del ojo).
- Fotofobia (intolerancia a la luz).

- Visión binocular o monocular.
- Campo visual corto.
- Situación visual es estable o progresiva.
- Dificultad para realizar un análisis de sombras, contornos, colores y movimientos.
- Distorsiones de la percepción que conduce a una interpretación errónea de la realidad.
- Dificultades en la atención e hiperactividad.
- Autoimagen alterada.
- Dificultades para imitar conductas, gestos y juegos.
- Presencia de miedos.

El rendimiento en las tareas visuales de las personas con debilidad visual depende de la concepción que tienen de sí mismas, ya que no son ciegas ni videntes, y en la mayoría de los casos no se puede determinar con exactitud cuánto ven, mucho menos lo que no ven, ni explicarlo a los demás. No tienen un parámetro para comparar su capacidad visual con la normalidad.

Los anteojos o lentes de contacto pueden mejorar el rendimiento de las personas con baja visión, pero no bastan para hacer que vean normalmente.

Una persona puede funcionar visualmente para algunas tareas y para otras no.

En ocasiones el resto visual no representa una ventaja si no todo lo contrario, puesto que no ve lo suficiente para manejarse como vidente y no maneja los instrumentos de los que podría beneficiarse una persona ciega rehabilitada. Esta situación implica un grado de tensión extra tanto física como psíquica lo cual puede generar otras patologías.

3.4.2 Agudeza visual

Capacidad para distinguir los objetos con nitidez a determinada distancia. La deficiencia se da cuando una persona no alcanza la unidad normal de agudeza visual (diez décimos o el conocido 20/200) y/o agudeza visual igual o inferior a 0'1 (1/10 de la escala Wecker) obtenida con la mejor corrección óptica posible.

3.4.3 Campo visual

Es la porción del espacio que alcanza a percibir una persona. La deficiencia se da cuando el CV se encuentra disminuido a 10° o menos. El campo visual de una persona con visión normal es de 180° en el campo visual horizontal en ambos ojos. 140° en el campo visual vertical en ambos ojos. 150° de percepción periférica con cada ojo.

3.4.4 Capacidad para distinguir la intensidad luminosa

Es la sensibilidad hacia los cambios de luz y la transformación de estos en impulsos eléctricos. Desde el punto de vista educacional se utiliza la siguiente clasificación.

Ciegos:
Personas que presentan una ausencia total de percepción visual y/o aquellas que, percibiendo o no luz, color y movimiento, no logran definir qué es o de dónde proviene y no pueden usar papel y lápiz para la comunicación escrita.

Débiles visuales:
Aquellos sujetos cuyos restos visuales (remanente visual) les permiten usar papel y lápiz para la comunicación escrita.

Débiles visuales profundos:
Son los que poseen una visión útil para realizar actividades de la vida diaria, pero tienen que utilizar en la escuela técnicas propias de los ciegos.

3.4.5 Deficiencias visuales:

A continuación, se describen las dos formas de deficiencia visual:
a) Pérdida de agudeza: se presenta en aquella persona cuya capacidad para identificar visualmente detalles está seriamente disminuida.
b) Pérdida de campo: la persona no percibe con la totalidad de su campo visual.

Todo esto antes mencionado será tomado en cuenta para el desarrollo del software y la creación de los ejercicios, sin dejar de lado las necesidades especificas que se tienen que cubrir según lo mencionado en el Centro de Estudios para Invidentes (CEIAC).

3.5 Tiflotecnología

La introducción de la informática en la vida general de las personas y en particular en el ámbito educativo supone en la mayoría de los casos una mayor facilidad en el acceso a la información, servicios y también, en el ámbito escolar y en la obtención de los conocimientos. Sin embargo, supondrá también un riesgo de exclusión si la persona no puede acceder al ordenador. Es fácil suponer la gran dificultad a la que se enfrenta la persona ciega y con baja visión para, precisamente, poder acceder a un medio donde la información se manifiesta fundamentalmente de forma gráfica y visual.

Afortunadamente, se ha desarrollado todo un campo de investigación y trabajo en torno a la consecución de la accesibilidad por parte de los discapacitados visuales. Todos aquellos dispositivos y programas, hardware y software, específicamente diseñados para hacer accesible a los ciegos la tecnología de la información se denominan «tiflotecnología».

La tiflotecnología basa su investigación y desarrollo en recursos que facilitan el acceso de las personas ciegas y deficientes visuales al ordenador, así como de aquellos dispositivos y sistemas autónomos con sus utilidades propias y específicamente desarrolladas para personas ciegas y deficientes visuales. Los ordenadores utilizados por las personas con discapacidad visual no son especiales, sino que son los usados por cualquier otra persona que no padezca ningún tipo de discapacidad; lo que sí son específicos son los periféricos que se instalan en ellos para que estas personas puedan acceder a la información. Los sistemas de acceso son diferentes dependiendo de que el usuario posea o no un resto visual aprovechable, ya que esto implicará que el sistema de acceso sea visual, táctil o parlante (síntesis de voz) (ONCE, s.f).

3.5.1 Análisis histórico de la Tiflotecnología - o tecnología para ciegos-

El término Tiflotecnología, del griego Tiflo (ciego), se incorpora al Diccionario de la Real Academia de la Lengua Española en 2008, donde se define como el "estudio de la adaptación de procedimientos y técnicas para su utilización por los ciegos".

Es difícil determinar una fecha concreta como origen de la Tiflotecnología, siendo quizás la Tiflomecánica e inventos que intentarán ayudar a los ciegos en este ámbito, los preludios de la misma. Se remonta entonces al primer prototipo de una máquina parlante en 1791 (Wolfgang von Kempelen 1734-1804), o a la aparición del Rafígrafo de Foucault en 1841 (Francois-Pierre Foucault) siguiendo una idea de Louis Braille (Roig 2000; Moreno-Montero, 2000).

A principios del siglo XX llega a España una de las primeras máquinas de escribir en Braille, una Picht de tecnología alemana (Oscar Picht 1871-1945). En 1939, en los laboratorios Bell, se implementa una síntesis espectral del habla compuesta por un oscilador. En 1962 surgen los primeros libros hablados, permitiendo escuchar textos a los ciegos que les narraban locutores o familiares en cintas abiertas. Uno de los primeros fue de la marca Clarcke & Smith y se reproducían gracias a sus 18 pistas en un sistema denominado Sistema A. En 1970 aparece la microelectrónica y en muy poco tiempo nace la Tiflotecnología en España (ONCE; 2009).

3.5.2 Cronología de la Tiflotecnología

En 1977 una estudiante ciega de matemáticas llamada Ángeles Ortiz, que necesitaba más recursos para superar su tesina universitaria, decide instruirse en una nueva tecnología que le permitirá poder ver con sus manos. El aparato en cuestión se denominaba "Optacon" (Telesensory), compuesto por una pequeñísima cámara que transmitía a 144 fototransistores de 24 filas por 6 columnas y que a su vez lo presentaba en una matriz táctil a 240 vibraciones por segundo, permitiendo que una persona ciega pudiese mediante el tacto, detectar cualquier imagen o texto. Este acontecimiento marca un punto de inicio en la historia de la Tiflotecnología en España (Ortiz, 2009).

Para 1978, en la Imprenta de la ONCE (Organización Nacional de Ciegos Españoles) en Madrid, se empezó a impartir a tres alumnos (José Luis Lorente, Ángel Martín y Leónides González) el primer curso de instructores para la distribución de unas 60 unidades del mencionado "Optacon" (Lorente, 2009).

En 1982 llegó a España el primer equipo de edición electrónica y de textos en braille denominado VersaBraille, compuesto por unos 20 caracteres dimorfos, una línea braille junto a una unidad central de proceso y para su gestión de datos utilizaba cintas de casete de 60 minutos. Sin duda un significativo avance para la época y probablemente uno de los primeros anotadores para ciegos.

Para 1983 se adquieren por la ONCE 40 unidades de lo que serían las primeras Telelupas, de la casa Philips y Teaman, un sistema que, mediante una cámara, proyectaba una señal de televisión de circuito cerrado aumentando la imagen o texto filmado, ayudándose de un zoom, y de este modo permitía a las personas con deficiencia visual acceder a letras e imágenes anteriormente inaccesibles

En 1984 se pone en marcha el primer sistema computarizado de producción braille en España, gracias al Negociado de Orientación Universitaria (NORE). Utilizaba un equipo de procesamiento de datos North Star de doble disquetera, con diez sectores en cada una, que redirigía la información a una impresora braille modelo LED 120 (120 caracteres por segundo

a una sola cara) de la compañía Triformation Systems (actualmente Enabling Technologies). Este conglomerado de hardware estaba comandado bajo un sistema operativo CPM y un software de manejo de trascripción de una firma muy importante entonces y que sigue existiendo hoy en día, denominada Duxbury Systems. Colaboraron en este proyecto Pedro Zurita, José Luis Lorente y José Sullivan.

Es en 1985 que nace la UTT (Unidad Tiflotécnica de la ONCE), un centro especial aplicado a la ceguera y deficiencia visual. El proyecto lo lidera en sus orígenes Luis Fenández Gosende, quien promueve un importante cambio en los existentes sistemas contables, de producción y de investigación de la empresa, e inicia un proyecto que no solo desarrollaría infinidad de nuevos accesos tiflotecnológicos, sino que también distribuiría e importaría diverso material, daría cobertura tecnológica y establecería una estrecha colaboración con la ONCE en sus adaptaciones al puesto de estudio, puesto de trabajo, departamento comercial, etc. En 2000 cambiará de nombre por CIDAT (Centro de Investigación, Desarrollo y Aplicaciones Tiflotecnológicas) (Cabrerizo, 2000).

Durante 1986 surgen en España dos nuevas creaciones de la popular Telesensory, las tarjetas Vista y VertPlus, una revolución para la época y que se implantaron en ese país hasta mediados de los años 90. Vista era una tarjeta que conseguía magnificar los 24x80 caracteres de una pantalla de ordenador, y que sólo será desbancada por el software de Zoomtext Plus en 1991. VertPlus fue una síntesis de voz algo más clara y comprensible que Infobox, gracias a la colaboración del catedrático español Muñoz Merino, y que se importa en España hasta el año 1996 (Arregui, 2004).

El primer software español dedicado a la tiflotecnología nace en 1988 con el nombre de COBRA. Este era un programa generador de dos necesarias funciones para la autonomía de las personas ciegas: por un lado, convertía textos ASCII a Braille; y por otro formateaba estos textos para poderlos mandar mediante puerto serie SR232 a una impresora Braille estándar. Colaboran en su desarrollo Aurelio Rodríguez de Data General y José Luis Lorente de UTT. Este programa marcará un antes y un después tanto en las producciones bibliográficas, como en la independencia del discapacitado visual, pudiéndose instalar en cualquier PC o anotador de

textos tipo Braille Hablado o posteriormente PC-Hablado, Sonobraille, etc. (Roig,2000;Moreno-Montero, 2000;Lorente, 2009).

En este año llega también Zoomtext a España, de AiSquared, uno de los magnificadores de pantalla más utilizados por los deficientes visuales en todo el mundo y que aparece ya en sus primeras versiones para MS-DOS, evolucionando rápidamente con Zoomtext Plus (1991) para Windows, Zoomtext Xtra (1997), etc. Las progresivas versiones se adaptaron a los avances del hardware y software, añadiendo ayudas de síntesis de voz (Level 2), u OCR (Level 3) en versiones posteriores.

A mediados de 1988 llega a los centros de investigación en España un pequeño equipo, que tras ser analizado y traducido su software al castellano, comienza la distribución de sus primeras unidades. Este producto, llamado Braille'n Speak o Braille Hablado, revolucionaría los métodos de escritura y almacenamiento de información para los ciegos. Aunque tuvo precedentes como el VersaBraille, el Delta, o el Eureka A4 de la australiana Robotrón Group, estos productos no tuvieron demasiado éxito por su escasa comprensión de su síntesis.

El Braille Hablado en cambio era pequeño, manejable y funcional, estaba compuesto por un procesador 8086 con 640Kb de memoria, un sistema operativo propio, teclado braille de 7 teclas y una síntesis de voz clara que se podía conectar a cualquier ordenador o impresora mediante puerto serie, con un software completo de gestión y manejo de datos, al que posteriormente se le sumarán complementos como el PCDisc de transferencia de datos desde el PC, el PCMaster de manejo de las funciones del mismo, así como una unidad de disco externa como periférico. Versiones posteriores del hardware como el Braille Hablado Plus o Braille Lite añadieron líneas braille de 18 o 40 caracteres. Tras 20 años de historia y 6000 unidades vendidas en España, todavía hay muchos ciegos que lo siguen utilizando (Arregui, 2004).

Gracias a un ingeniero catalán de telecomunicaciones, Andrés Ursueguía Balbuena, se creará en 1990 el denominado CIBERVEU o CIBERVOZ, la primera síntesis de voz fabricada en España y dedicada a apoyar a las personas ciegas en el manejo de los ordenadores. Se desarrolló con el fin de competir con las ya existentes VertPlus o Infobox, buscando añadir más comprensión y variedad de lenguas (como el catalán) frente a los citados productos. A este invento le seguirán versiones mejoradas como el CIBER 232, CIBER 232P, CIBERGUIA, etc. Andrés Ursueguía

seguirá creando otras patentes como el "Sistema de Asistencia a la Orientación para Invidentes 1994" junto a Armando Lázaro, buscando constantemente soluciones hacia la disminución de barreras y desarrollando avances en la Tiflotecnología en España (CC&P, 1998).

En 1992 nacía el Habla, uno de los programas más sencillos, eficaces y utilizados en España, fruto de un entusiasta instructor tiflotécnico de la delegación de Murcia de la ONCE, Cayetano Meroño, que en 1992 se da cuenta gracias a un alumno de que los sintetizadores existentes no distinguían suficientemente a nivel fonético algunas letras como la 'p' y la 'd'. Decide entonces ponerse a trabajar en un programa que mediante ensamblador redirija correctamente los elementos conflictivos al Braille Hablado. Un año más tarde la ONCE lo empieza a distribuir oficialmente, extendiéndose posteriormente con diferentes versiones para todos los sintetizadores de voz, anotadores existentes y ordenadores compatibles

Tras un año en pruebas, en noviembre de 1993 nace "Tiflotecnia", el primer programa radiofónico dedicado a la divulgación pedagógica de la tiflotecnología (UTT, 1993). De manos de dos deficientes visuales, Enrique Varela y Eugenio Martín, se empieza a emitir en Canal 11, la radio interna de la ONCE, dedicado a difundir toda la información relevante a las nuevas y cada vez más difundidas tecnologías para ciegos y deficientes visuales. Este espacio radiofónico daba la oportunidad de escuchar, por ejemplo, cómo hablaban esos novedosos anotadores, cómo se comportaban las últimas síntesis de voz o magnificadores de pantalla en sus diferentes configuraciones. Tras unos años y diferentes cambios de nombre (la Disquetera, El Tren de la Informática y El Tren 2.0), dejó de emitirse en 2004 con el cierre de Canal 11.

Tras varios meses de desarrollo y pruebas, se empieza en 1994 a distribuir el primer anotador parlante desarrollado en España, denominado PC-Hablado, consistente en un procesador 80286, 2Mb de memoria de almacenamiento más una tarjeta PCMCIA de 1Mb adicional. Como sistema de escritura utilizaba un teclado braille, un sistema operativo MS-DOS 5.x, una síntesis de voz italiana de la firma Audiology junto con una evolución compilada del Habla, y diferentes aplicaciones como procesador de textos, agenda, calculadora, etc. Su mayor artífice e investigador fue Rodrigo Roldan. Posteriormente el PC-Hablado evolucionará, surgiendo Sonobraille en 1998 (Arregui, 2004) (UTT, 1995).

El 13 de Diciembre de 1995 nace Tiflonet (http://usuarios.lycos.es/tiflonet/), la primera Web dedicada a la Tiflotecnología en España, creada por Enrique Varela "como apoyo a sus investigaciones y punto de encuentro para el creciente número de usuarios ciegos que iban apareciendo". Tras dos años (16 de abril de 1997) crea la primera lista de distribución de habla hispana sobre tiflotecnología, y el 15 de abril de 1998 se inaugura Tiflolupa, el buscador accesible en castellano para ciegos. En el cuarto aniversario de su creación, Oscar Gorri releva a Enrique Varela en la administración de los servicios de la lista y página web.

Tiflowin es el primer lector de pantalla creado en España para PC y compatibles, que funcionará con una síntesis de voz o línea braille bajo Windows 3.x. Permite trabajar mediante dos modos de operación: información en tiempo real mediante el seguimiento de foco, y exploración / navegación de los elementos en pantalla. En su desarrollo se produce una colaboración entre Microsoft, ONCE, CETTICO y VAUM Electronic, que culminará con la creación de las librerías SAPI y BAPI que actualmente todavía se utilizan en diferentes productos y síntesis de voz. Este proyecto evolucionará desarrollando por parte de ONCE el Lector 98 sin demasiado éxito. Tiflowin se dejó de producir en 2001.

Es en 1998, en Cantabria donde se crea la primera asociación de usuarios de Tiflotecnología denominada UTLAI (Usuarios de Tiflotecnología para el Libre Acceso a la Información), por Soledad Mochales, buscando estándares en la tecnología para ciegos y accesibilidad en la red, persiguiendo el apoyo y ayuda entre usuarios, y publicando una revista periódica. En diciembre de 2007 UTLAI y TIFLOCLUB (una popular web y lista de distribución creada tras Tiflonet por Enrique Varela), se juntan aunando esfuerzos y recursos.

En 1999 Se distribuye en España el lector de pantalla Jaws for Windows de Henter Joice (posteriormente Freedom Scientific tras unirse a Blazzie Engineering y Arquenstone en abril de 2000). Este software, durante la última década ha llegado a ser el lector de pantalla más utilizado por los usuarios ciegos de todo el mundo, superando rápidamente a toda su competencia en el mercado (Tiflowin, Lector 98, Home Pagre Reader de IBM, etc). A finales de 2007 surge NVDA, que por su cualidad de freeware e interesantes características quizá con el tiempo pueda alcanzar una popularidad similar (Arregui, 2004) (Freedom Scientific, 2009).

En el 2003, Mobile Accessibility de Code Factory, el primer lector de pantalla para teléfonos móviles fabricado en España, que irá evolucionando hacia el Mobile Speak 2004, Mobile Magnifier 2005 (con magnificador de pantalla para personas con deficiencia visual) o Mobile Magnifier Lite 2007 (Code Factory; 2009). Posteriormente saldrían versiones para Pocket PC, SmartPhone, etc. De otras compañías foráneas surgieron productos como Talks (software) u Oasis (teléfono específico para ciegos) que aunque llegaron a al país, no alcanzaron el mismo éxito.

En diciembre de 2008 las normas WCAG 2.0 dejan de ser beta, dando un paso adelante casi una década después de sus predecesoras. Emmanuelle Gutiérrez, tal y como lo hizo en su primera versión, participa también en los grupos de trabajo de estas directrices. Unos meses después los diferentes gestores de evaluación de páginas web (TAW, HERA) se actualizarán a las mismas.

En 2009 se lanza Kapten de Kapsys, un GPS portátil que funciona por voz, suponiendo un antes y un después en la autonomía de las personas ciegas. Se trata de un aparato muy reducido en tamaño y peso, con una base de datos de TeleAtlas, antena de alta sensibilidad SRIF Star III, síntesis de voz, reconocimiento de la misma, reproductor MP3, y que funciona sobre el sistema operativo Linux. Este GPS permite a los ciegos llegar a un destino desconocido sin ayuda de terceros.

En la actualidad podemos encontrar gran variedad de equipos adaptados que incorporan algún programa para que puedan ser usados por personas con discapacidad visual. Gracias a estas adaptaciones el usuario solo tendrá que conocer el aparato que está utilizando y los distintos comandos del programa que le proporcione la adaptación.

La principal ventaja de usar este tipo de equipos es que cuando se tiene cualquier problema siempre se puede solicitar ayuda a cualquier persona, debido a que ésta siempre podrá echar un vistazo a la pantalla y resolverlo.

A continuación, se mencionan algunos equipos adaptados que están siendo utilizados actualmente:

Lectores de pantalla

Los lectores de pantalla son aplicaciones para identificar e interpretar aquello que se muestra en pantalla. Esta interpretación se representa a continuación al usuario mediante sintetizadores de texto a voz, iconos sonoros, o una salida braille.

Magnificadores de pantalla

Los magnificadores de pantalla son programas para la accesibilidad que permiten ampliar los caracteres y configurar los colores dependiendo de la necesidad que posea el usuario.

Teléfonos y PDAs

Existen otros equipos para los que se han desarrollado magnificadores y lectores de pantalla.
La empresa española Code Factory ha desarrollado los siguientes productos:

- Mobile speak. Lector de pantalla para teléfonos móviles.
- Mobile magnifier. Magnificador de pantalla para teléfonos móviles.
- Mobile speak pocket. Lector de pantallas para PDAs que utilicen Windows Mobile.

Además, existe otro lector de pantalla para teléfonos móviles denominado Talks, aunque es únicamente para la plataforma Symbian y para terminales con teclado, ya que las pantallas resistivas no ofrecen precisión de gestos. Es necesario mencionar la inclusión del Voice Over de Apple en sus terminales táctiles a partir del modelo 3GS, que los hace cien por cien operativos para personas ciegas y con baja visión gracias a una serie de gestos que permiten interactuar con la terminal y utilizar la totalidad de sus funciones.

Hasta aquí se ha presentado un recorrido histórico de la tiflotecnología, destacando los avances más significativos de las últimas décadas, e intentando contribuir a una mayor comprensión sobre las dificultades y barreras con las que los usuarios ciegos y deficientes visuales tienen que enfrentarse en el acceso a la información, así como de la necesidad de que toda nueva tecnología sea usable para todos. Cabe mencionar que el presente proyecto utiliza la tiflotecnología haciendo uso de la computadora como medio para el diseño, desarrollo e implementación del software que a su vez estará siendo utilizado en una computadora para dar la terapia de estimulación visual.

3.6 Herramientas, lenguajes y tecnologías que se emplearán en el desarrollo del Software

3.6.1 IDE

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

Normalmente un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse; otros no, tales como SharpDevelop y Lazarus.

El límite entre un IDE y otras partes del entorno de desarrollo de software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos. (Ramos, Lozano, María, 2000).

3.6.2 API

La abreviatura viene del término en inglés Application Programming Interfaces (Interfaces de programación de aplicaciones), las API son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas, ya que no tiene que «escribir» códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa.

3.6.3 NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

Lenguajes de programación usados en NetBeans:

- Java
- C++
- PHP
- Ruby
- y otros más.

NetBeans está desarrollado en Java y por eso requiere la instalación del JDK y la JVM para su funcionamiento.

Este tipo de software proporciona al programador herramientas para que el desarrollo de software sea lo menos complicado, incluyendo utilidades como resaltado y autocompletado de código, cómpila y ejecuta el código que se crea además con esta IDE se pueden diseñar interfaces de una manera mucho más sencilla.

Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (Netbeans sitio oficial, s.f.).

3.6.4 HTML

Sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros), este no se incrusta directamente en su código, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

3.6.5 HTML5, CCS3 y JavaScript

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, ni siquiera una mejora de esta ya antigua tecnología, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red.

HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. HTML provee los elementos estructurales, CSS3 se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y JavaScript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales.

CSS3 es un lenguaje utilizado en la presentación de documentos HTML. Este lenguaje sirve para organizar la presentación y aspecto de una página web, es principalmente utilizado por parte de los navegadores web de internet y por los programadores web informáticos para elegir multitud de opciones de presentación como colores, tipos y tamaños de letra, etc.

JavaScript es un lenguaje de programación que se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web. Los navegadores modernos interpretan el código JavaScript integrado en las páginas web (Gauchat, 2012).

3.6.6 Metodologías de desarrollo de software

¿En qué consisten las Metodologías de desarrollo de software?

Una Metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Regularmente este tipo de metodología, tienen la necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto, comprendan perfectamente la metodología y en algunos casos el ciclo de vida del software que se pretende seguir.

Modelo Incremental:

El desarrollo incremental se basa en la idea de diseñar una implementación inicial, exponer ésta al comentario del usuario, y luego desarrollarla en sus diversas versiones hasta producir un sistema adecuado.

El modelo incremental ejecuta una serie de avances, llamados incrementos, que en forma progresiva dan más funcionalidad al cliente conforme se le entrega cada versión del sistema.

El modelo de proceso incremental se centra en que en cada incremento se entrega un producto que ya opera. Los primeros incrementos son versiones desnudas del producto final, pero proporcionan capacidad que sirve al usuario y también le dan una plataforma de evaluación.

El desarrollo incremental es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio (figura 3.1).

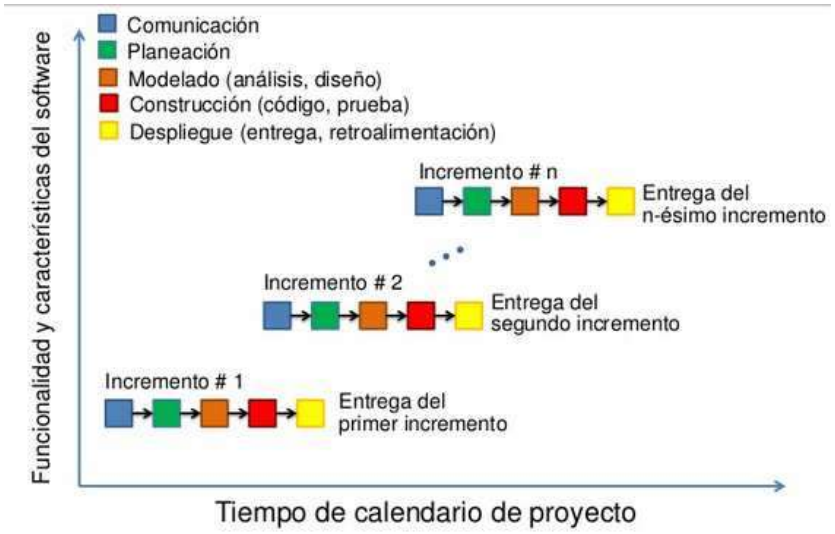

Figura 3.1 Modelo Incremental.

Modelo Iterativo:

Consiste en la iteración o repetición de varios ciclos de vida en cascada. Al final se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto, éste es quien evalúa el producto proponiendo cambios para mejora. Este modelo consiste en hacer una primera versión del software, para posteriormente ir elaborando otras nuevas que se vayan acercando más al sistema final (figura 3.2).

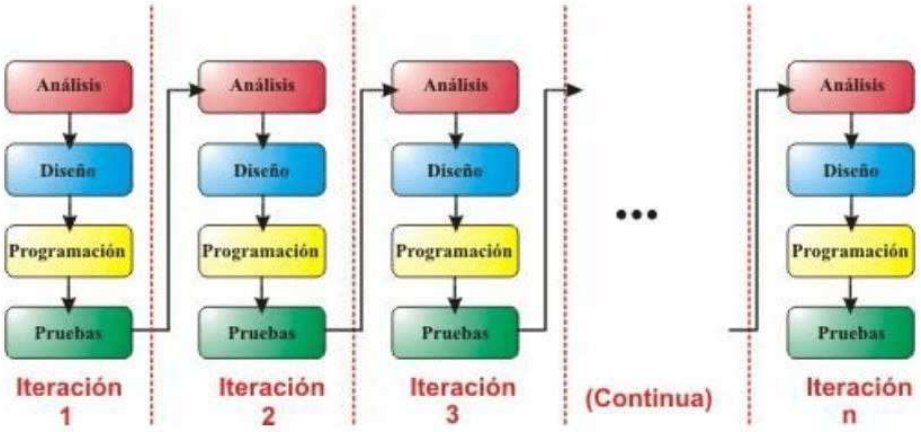

Figura 3.2 Modelo Iterativo.

IV. DESARROLLO DEL SOFTWARE

Para el desarrollo de software se utilizaron las metodologías de desarrollo Incremental e iterativo puesto que con estás es posible realizar modificaciones o actualizaciones del software y en cada iteración o nueva versión se puede ir aumentando la funcionalidad e ir mejorando la calidad respecto a la versión anterior del software, permitiendo que el usuario vaya de la mano con el proyecto durante la realización y de esta manera poder proporcionarle un resultado óptimo.

Con estos modelos de desarrollo con cada iteración se generará un prototipo cada vez mejor. La ventaja de los prototipos es que se pueden tener guardados por si en determinado momento se desea volver atrás, pues a diferencia del modelo de cascada se puede retroceder cuando se requiera y los prototipos sean necesarios volver a utilizar una y otra vez.

4.1. Análisis

Se realizó el proceso de análisis del proyecto en el Centro de Estudios para Invidentes A.C (CEIAC) asistiendo a entrevista con el encargado del área de Tiflotecnología el Ingeniero Fernando Beltrán. También se asistió a la terapia que toman los niños con deficiencias visuales, de lo observado de la terapia y de la información proporcionada de la terapeuta que ayuda a los niños con deficiencias visuales se recabaron datos útiles para el análisis.

Análisis de Requerimientos

El Centro de Estudios para Invidentes A.C. solicita un software que sirva de apoyo en la terapia de estimulación visual de niños con baja visión. Dentro de los requerimientos técnicos se solicita que el software este diseñado para computadora y que el programa se realice con HTML ya que de esta manera se puede ejecutar en cualquier navegador y con ello se facilite su manejo y utilización. Se solicita que el software utilice figuras y dibujos animados actuales para que sean atractivos y motiven al niño al momento de tomar la terapia. Además se solicita se le pueda agregar al software un control donde se pueda aumentar y disminuir la velocidad con la que se despliegan las animaciones.

A continuación, se describen los procesos a detalle que van a formar parte del software:

Menú principal

El terapeuta inicia la sesión entrando directamente al menú principal del sistema. El sistema muestra dentro del menú los ejercicios donde el terapeuta podrá seleccionar el ejercicio interesado para la terapia de estimulación visual (figura 4.1).

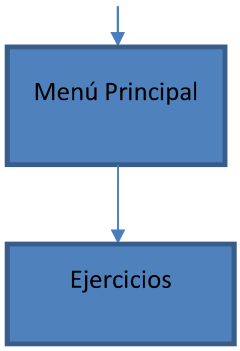

Figura 4.1 Menú principal.

Seleccionar los ejercicios

En esta parte se pueden seleccionar dentro de un menú desplegable los diferentes ejercicios que la terapeuta puede utilizar durante la terapia de estimulación visual con los niños del CEIAC (figura 4.2).

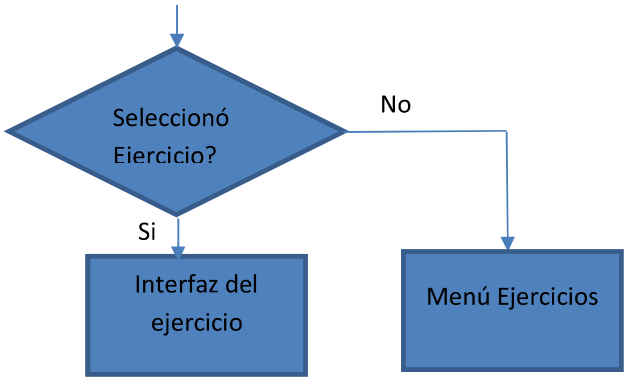

Figura 4.2 Selección de ejercicios.

Interfaz del ejercicio

En este apartado se muestra la interfaz del ejercicio que previamente el terapeuta ya seleccionó y es con la cual el niño con baja visión estará trabajando. Aquí la terapeuta es la encargada de seleccionar el color del fondo del ejercicio, el color de la animación y la velocidad con la que se despliega la animación (figura 4.3).

.

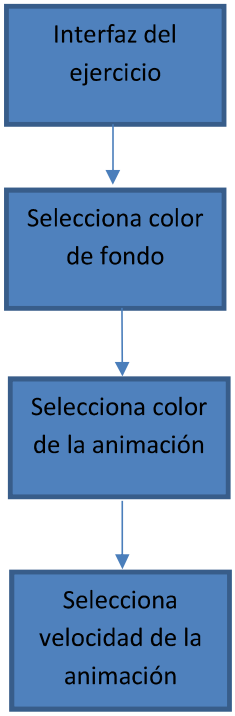

Figura 4.3 Interfaz del ejercicio.

4.2. Diseño

En esta fase se define la funcionalidad del software y se diseñan las interfaces de los ejercicios.

El software tiene un menú principal, en este menú se pueden seleccionar los ejercicios, dentro de cada uno de los ejercicios, se elijen ciertos parámetros antes de ejecutarse la animación. Por ejemplo, se selecciona el color del fondo de los ejercicios, se seleccionan el color de la animación, también se selecciona la velocidad con que se despliegan las animaciones. Después de haber seleccionado los parámetros se ejecuta la animación y es entonces cuando el niño estará trabajando con el ejercicio.

Descripción de procesos.

A continuación, se muestra la descripción de cada proceso del sistema. Se presentan los casos de uso de las pantallas y ejercicios del software.

Caso uso: Menú principal

Actores: Terapeuta

Resumen: La terapeuta accede al menú principal donde se encuentran los accesos a cada uno de los ejercicios. El terapeuta da clic sobre el ejercicio para seleccionarlo y poder entrar y usarlo en la terapia con el niño. Tal como se muestra en la tabla 4.1 Acciones y respuesta del sistema. Menú principal y en las figuras 4.4 Pantalla principal y 4.5 Menú de los ejercicios.

Tabla 4.1 Acciones y respuesta del sistema. Menú principal.

Acciones	Respuesta del Sistema
1. La terapeuta inicia entrando a la pantalla principal del sistema. 3. El terapeuta ingresa al menú de ejercicios. 5. El terapeuta selecciona el acceso al ejercicio interesado.	2. El sistema muestra el menú de los ejercicios. 4. El sistema muestra los accesos a cada uno de los ejercicios.

Figura 4.4 Pantalla principal.

Figura 4.5 Menú de los ejercicios.

Caso uso: Configuración de la Animación Figuras.

Actores: Terapeuta, niño, sistema.

Resumen: El terapeuta ingresa al ejercicio y selecciona el color del fondo, el color de la línea y la figura que se mostrará en la animación. Después de haber hecho las configuraciones del ejercicio, se inicia la animación con la cual el niño estará trabajando, tal como se muestra en la tabla 4.2 Acciones y respuestas del sistema. Configuración de la Animación Figuras. Y en la figura 4.6 Animación Figuras.

Tabla 4.2 Acciones y respuesta del sistema. Configuración de la Animación Figuras.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta selecciona el color del fondo del ejercicio, el color de la animación y la velocidad con la que se despliega la animación. 5. El terapeuta inicia el ejercicio.	2. El sistema muestra las opciones de configuración del ejercicio. 4. El sistema hace las configuraciones del ejercicio. 6. El sistema ejecuta la animación.

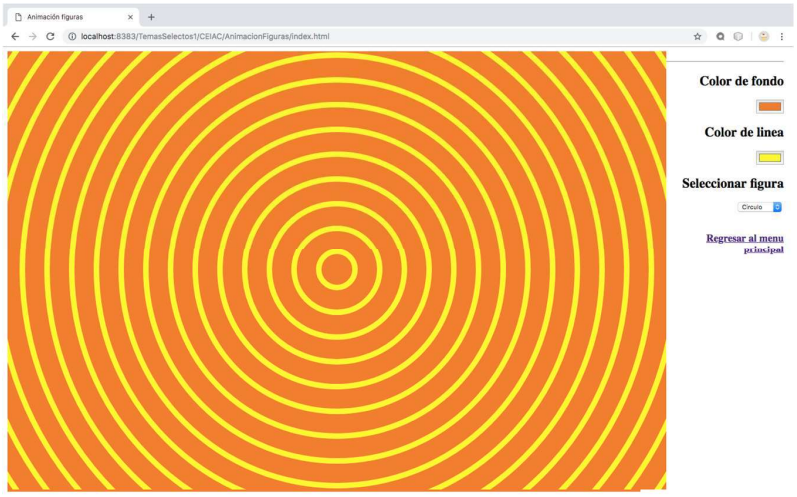

Figura 4.6 Animación Figuras.

Caso uso: Configuración de la Animación Pelotas.

Actores: Terapeuta, niño, sistema.

Resumen: En este ejercicio existe una pelota que tendrá un color diferente al resto de las que se encuentran en la animación. El terapeuta deberá seleccionar el color de la pelota, además de seleccionar el número de pelotas que van a aparecer en la animación. En este ejercicio se trabaja con el estimulo de rastreo en la terapia, ya que el niño debe de ser capaz de seguir la pelota que es de diferente color, tal como se muestra en la tabla 4.3 Acciones y respuesta del sistema. Configuración del la Animación Pelotas y en la figura 4.7 Animación Pelotas.

Tabla 4.3 Acciones y respuesta del sistema. Configuración de la Animación Pelotas.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta selecciona el color de la pelota y el número de pelotas que van a parecer en la animación.	2. El sistema muestra las opciones de configuración del ejercicio. 4. El sistema hace las configuraciones del ejercicio y ejecuta la animación.

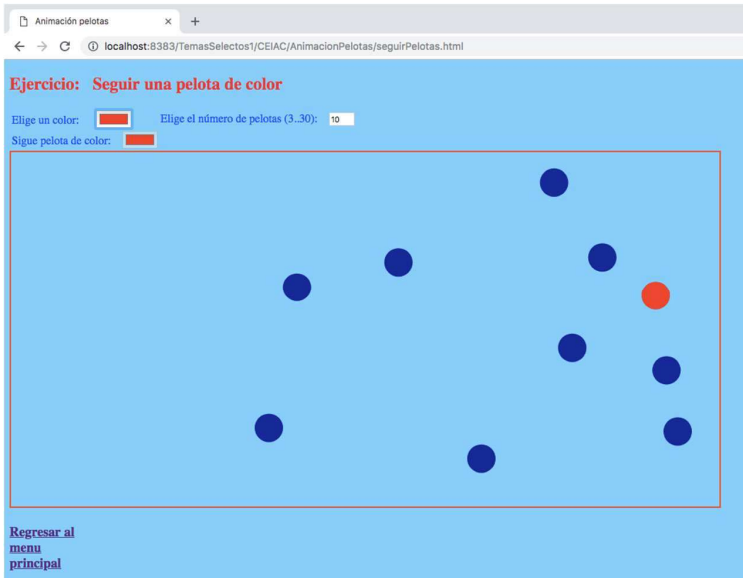

Figura 4.7 Animación Pelotas.

Caso uso: Configuración de la animación Causa-Efecto.

Actores: Terapeuta, niño, sistema.

Resumen: El terapeuta ingresa al ejercicio y selecciona una imagen en el menú desplegable, selecciona la velocidad con la que se va a desplegar la animación, y por último selecciona el color del fondo que se mostrará en el ejercicio. Después de haber realizado las configuraciones del ejercicio, se inicia la animación dando clic en el botón iniciar, tal como se muestra en la tabla 4.4 Acciones y respuesta del sistema. Configuración de la Animación Causa-Efecto. Y en la figura 4.8 Animación Causa-Efecto.

Tabla 4.4 Acciones y respuesta del sistema. Configuración de la animación Causa-Efecto.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta selecciona la imagen, la velocidad y el color del fondo de la animación.	2. El sistema muestra las opciones de configuración del ejercicio. 4. El sistema hace las configuraciones del ejercicio y ejecuta la animación.

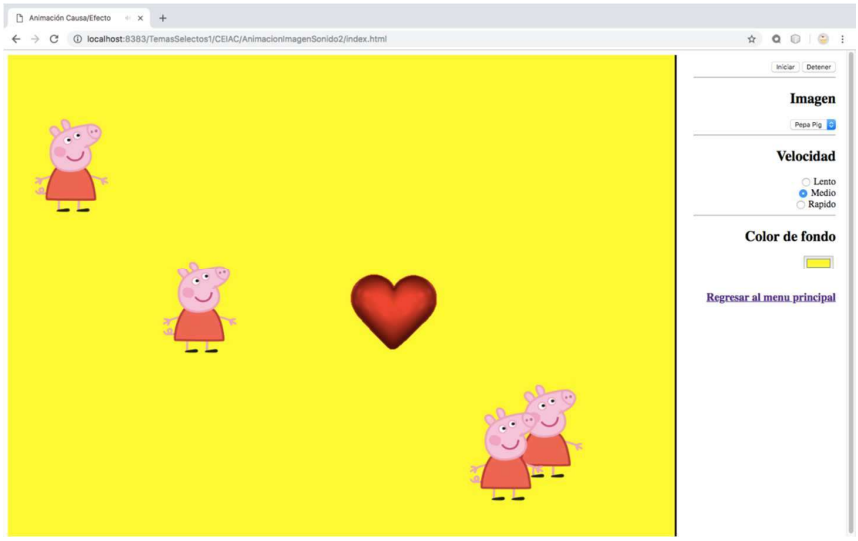

Figura 4.8 Animación Causa-Efecto.

Caso uso: Animación Atrapar.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema automáticamente carga la animación para que el niño pueda interactuar directamente con ella siempre guiado de la terapeuta. Durante este ejercicio las animaciones están moviéndose en diferentes direcciones y es entonces cuando el niño debe de atraparlas. En este ejercicio se está trabaja con el estímulo de rastreo, así como se muestra en la tabla 4.5 Acciones y respuesta del sistema. Animación Atrapar. Y en la figura 4.9 Animación Atrapar.

Tabla 4.5 Acciones y respuesta del sistema. Animación Atrapar.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da clic sobre la animación para que empiece el ejercicio.	2. El sistema muestra la animación lista para ejecutarse. 4. El sistema empieza la animación.

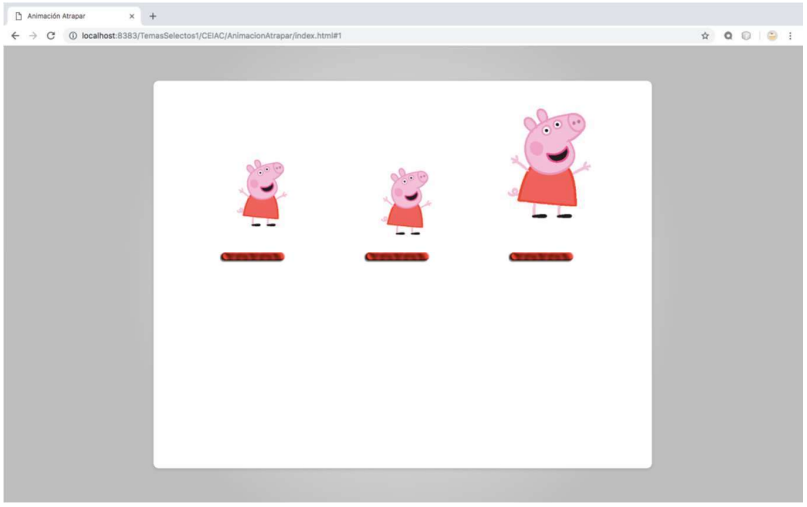

Figura 4.9 Animación Atrapar.

Caso uso: Animación Paletas.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema automáticamente carga la animación para que el niño pueda interactuar con ella siempre guiado por la terapeuta. En este ejercicio las animaciones de desplazan de arriba hacia abajo, es entonces cuando el niño debe de ser capaz de dar clic sobre la figura antes de que topen con el piso y se estrelle la paleta. En este ejercicio se está trabajando con el estímulo de rastreo, tal como se muestra en la tabla 4.6 Acciones y respuesta del sistema. Animación Paletas. Y en las figuras 4.10 Animación Paletas primera parte, 4.11 Animación Paletas segunda parte, 4.12 Animación Paletas tercera parte.

Tabla 4.6 Acciones y respuesta del sistema. Animación Paletas.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da indicaciones al niño de cómo interactuar con el ejercicio.	2. El sistema ejecuta la animación automáticamente.

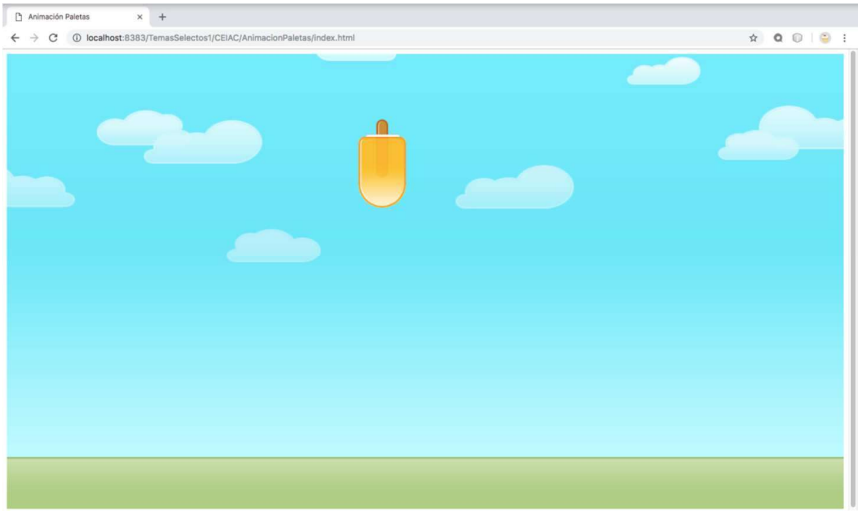

Figura 4.10 Animación Paletas primera parte.

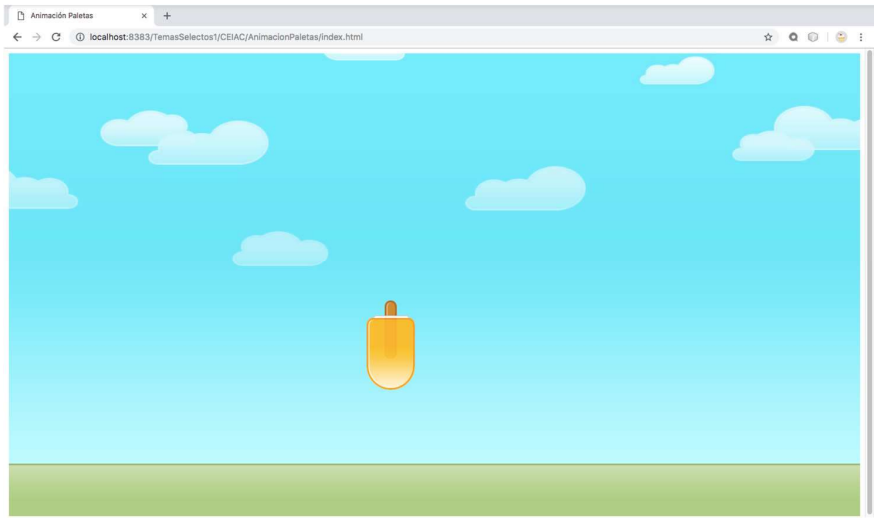

Figura 4.11 Animación Paletas segunda parte.

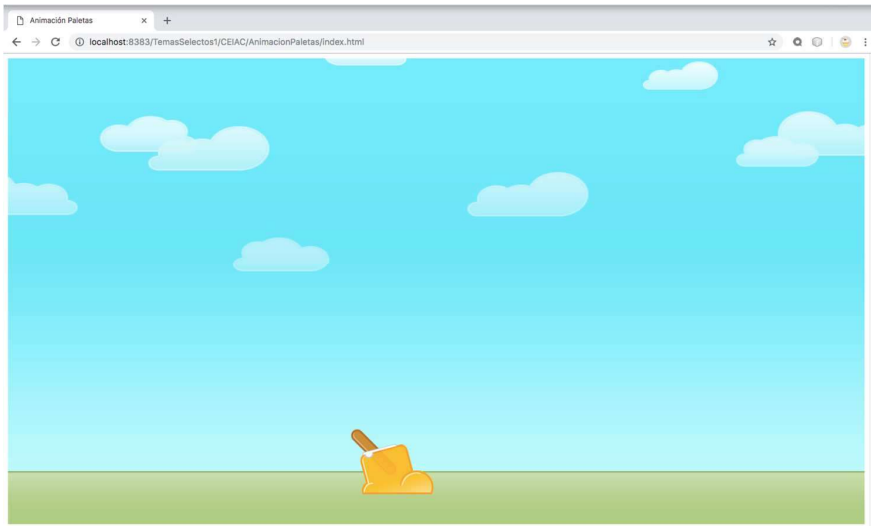

Figura 4.12 Animación Paletas tercera parte.

Caso uso: Animación Gotas.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema solicita dar clic para iniciar la animación. Después de haberle dado clic se ejecuta automáticamente la animación y el niño deberá observar cómo van mostrándose las figuras. En todo momento el terapeuta deberá guiar al niño mientras se ejecute la animación, tal como se muestra en la tabla 4. 7 Acciones y respuesta del sistema. Animación Gotas. Y en la figura 4.13 Animación Gotas.

Tabla 4.7 Acciones y respuesta del sistema. Animación Gotas.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da indicaciones al niño de cómo interactuar con el ejercicio.	2. El sistema solicita dar clic para iniciar la animación. 4. El sistema ejecuta la animación automáticamente.

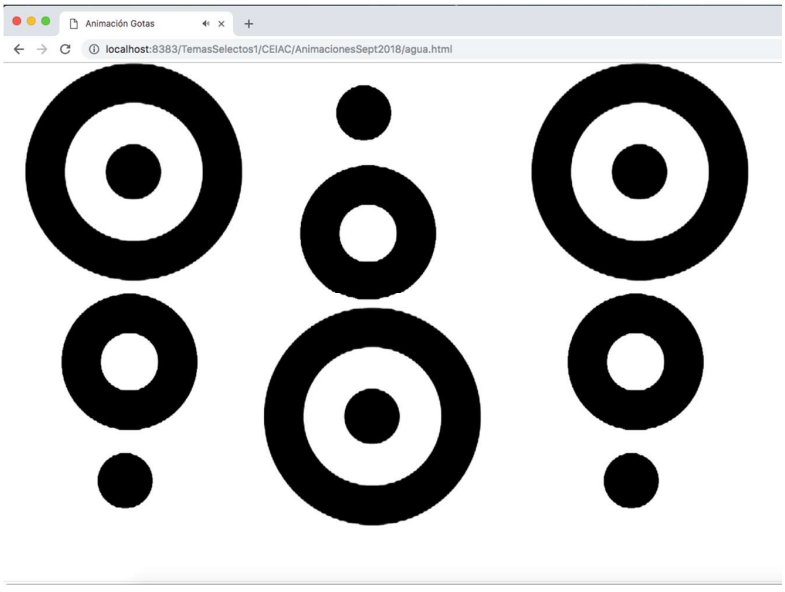

Figura 4.13 Animación Gotas.

Caso uso: Animación Cilindro.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema automáticamente carga la animación para que el niño pueda interactuar con ella siempre guiado de la terapeuta. En este ejercicio la animación se ejecuta una y otra vez, el niño deberá de ser capaz de seguir la animación que está en movimiento, tal como se muestra en la tabla 4.8 Acciones y respuesta del sistema. Animación Cilindro. Y en la figura 4.14 Animación Cilindro.

Tabla 4.8 Acciones y respuesta del sistema. Animación Cilindro.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da indicaciones al niño de cómo interactuar con el ejercicio.	2. El sistema ejecuta la animación automáticamente.

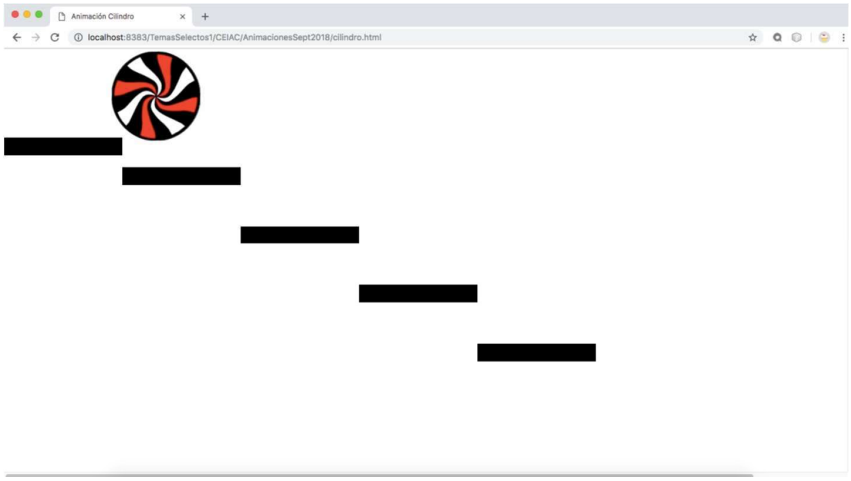

Figura 4.14. Animación Cilindro.

Caso uso: Animación Círculos.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema automáticamente carga la animación para que el niño pueda interactuar con ella. En este ejercicio el niño deberá ser capaz de dar clic sobre los círculos. De esta manera con el ejercicio está trabajando con el estímulo causa y efecto ya que cuando el niño da clic sobre el círculo se muestra el efecto del cambio del color del mismo, tal como se muestra en la tabla 4.9 Acciones y respuesta del sistema. Animación Círculos. Y en la figura 4.15 Animación Círculos.

Tabla 4.9. Acciones y respuesta del sistema. Animación Círculos.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da indicaciones al niño de cómo interactuar con el ejercicio.	2. El sistema ejecuta la animación automáticamente.

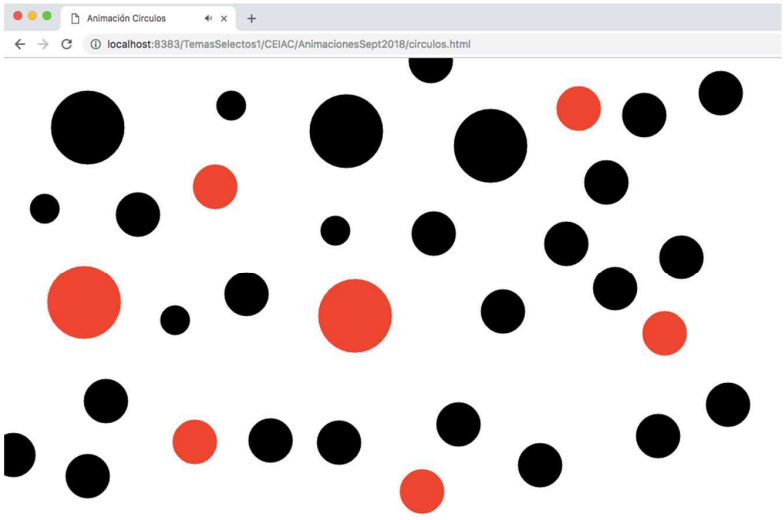

Figura 4.15 Animación Círculos.

Caso uso: Animación Líneas.

Actores: Terapeuta, niño, sistema.

Resumen: El sistema automáticamente carga la animación para que el niño pueda interactuar con ella, en este ejercicio la animación se va mostrando de izquierda a derecha, el niño deberá de observar cómo se va desplegando la animación. En este ejercicio se está trabajando con el estímulo de discriminación visual, ya que el niño debe de ser capaz de identificar los colores de los bloques que se muestran en la animación, tal como se muestra en la tabla 4.10 Acciones y respuesta del sistema. Animación Líneas. Y en las figuras 4.16 Animación Líneas primera parte, 4.17 Animación Líneas segunda parte, 4.18 Animación Líneas tercera parte, 4.19 Animación Líneas cuarta parte.

Tabla 4.10 Acciones y respuesta del sistema. Animación Líneas.

Acciones	Respuesta del Sistema
1. La terapeuta ingresa al ejercicio. 3. La terapeuta da indicaciones al niño de cómo interactuar con el ejercicio.	2. El sistema ejecuta la animación automáticamente.

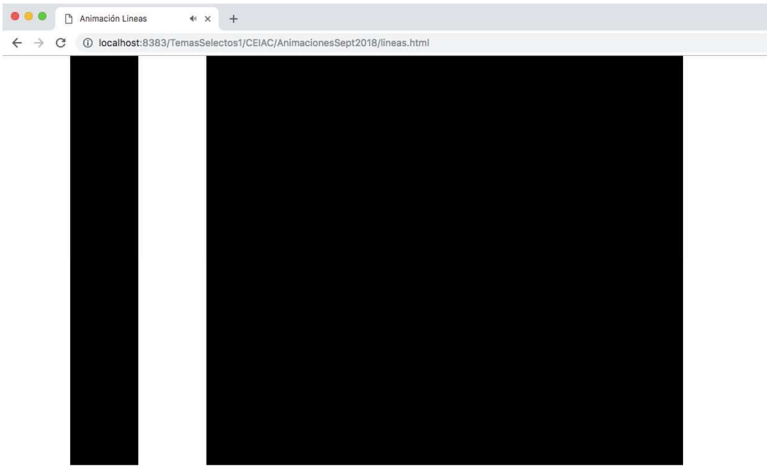

Figura 4.16 Animación Líneas primera parte.

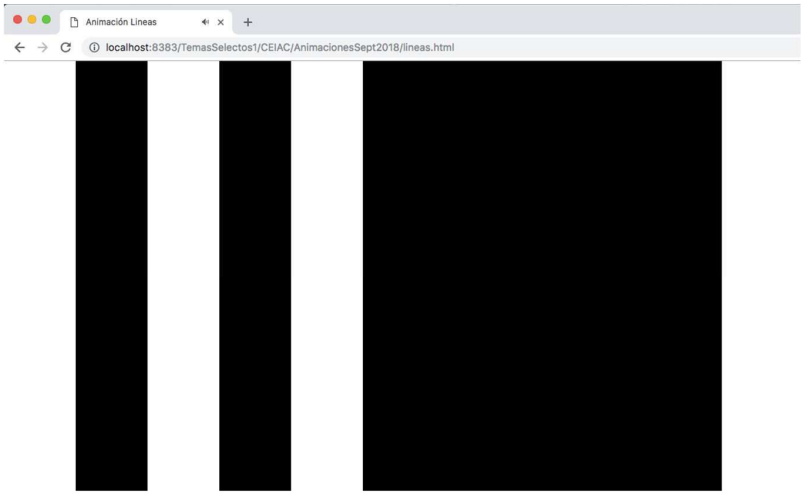

Figura 4.17 Animación Líneas segunda parte.

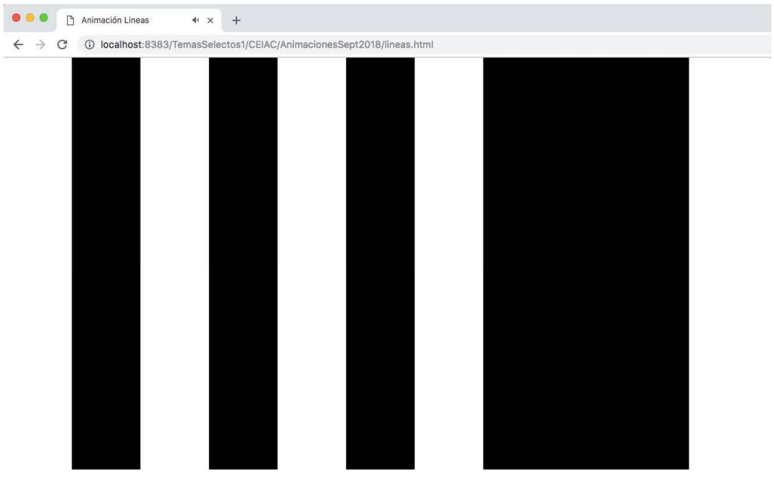

Figura 4.18 Animación Líneas tercera parte.

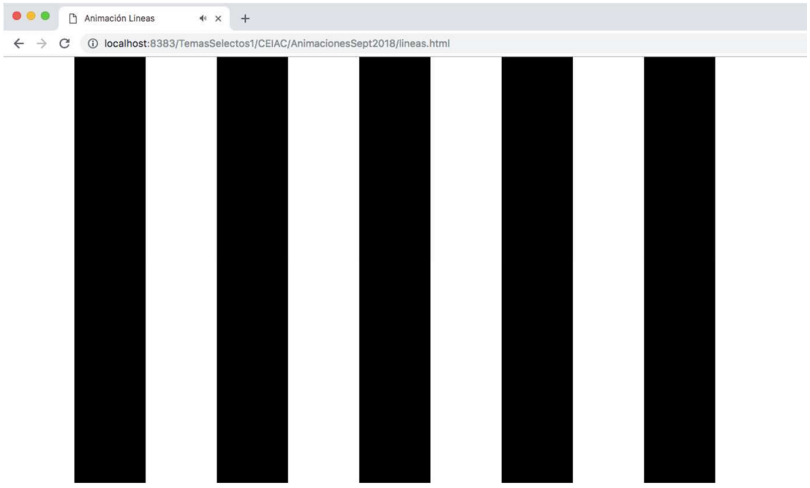

Figura 4.19 Animación Líneas cuarta parte.

4.3 Implementación.

Para la realización del proyecto se utilizó el entorno de desarrollo integrado (IDE) NetBeans ya que este permite la creación de archivos HTML5 y CSS además que se puede usar JavaScript donde se pueden crear efectos atractivos y dinámicos en las aplicaciones web. Se utilizaron algunas API's que incorpora HTML5 las cuales permiten generar y trabajar con gráficos en la pantalla, crear animaciones o manipular imágenes y videos. Esto último fue muy útil para el desarrollo proyecto ya que el sistema requiere manejo de imágenes y animaciones en los ejercicios de estimulación visual usado con los niños con deficiencias visuales del CEIAC.

En el diseño se muestran algunas de las principales pantallas implementadas, en los anexos esta el código de la ventana de inicio del sistema y menú, el de la animación de pelotas, el de la animación de atrapar a Pepa, el de la animación causa y efecto, el de la animación de paletas, el

de la animación de gotas, el de la animación del cilindro, el de la animación de círculos y el de la animación de líneas.

4.3.1 Equipo (hardware)

Para el desarrollo del sistema se utilizó una computadora portátil marca Apple, MacBook Air con procesador 1.3 GHz Intel Core i5, 4 GB de memoria RAM 1600 MHz DDR3, Disco Duro de 120 GB, pantalla de 13.3 pulgadas y sistema operativo OS X El Capitán.

4.4 Pruebas

En esta sección se comentan los resultados de las pruebas del software para asegurarnos del correcto funcionamiento.

4.4.1 Pruebas de los ejercicios

Se llevaron a cabo las pruebas de los ejercicios que contiene el software. Se probaron los ejercicios utilizándolos en diferentes sistemas operativos y diferentes navegadores. Además se probó la funcionalidad, así como que los ejercicios cumplieran con los objetivos planteados previamente en el primer capítulo. El resultado de las pruebas fue que cargaron y se ejecutaron correctamente los ejercicios en los navegadores en que se probaron. Los navegadores en que se probó el sistema son: Google Chrome versión 7.0, también se probó en el navegador Safari versión 9.1.

En las figuras 4.20, 4.21 y 4.22 podemos ver a una niña tomar terapia de estimulación visual en el Centro de estudios para invidentes, la niña guiada de la terapeuta interautó con las animaciones círculos, paletas y causa-efecto, asi como este ejemplo, hay varios niños mas que tomaron la terapia de estimulación visual con estas animaciones y el resultado fue que los niños en todo momento se sintieron motivados con los ejercicios, además de que utilizaron los estímulos de rastreo, causa-efecto y profundidad los cuales cumplen con el objetivo planteado en un principio en ayudar a estimular su visión.

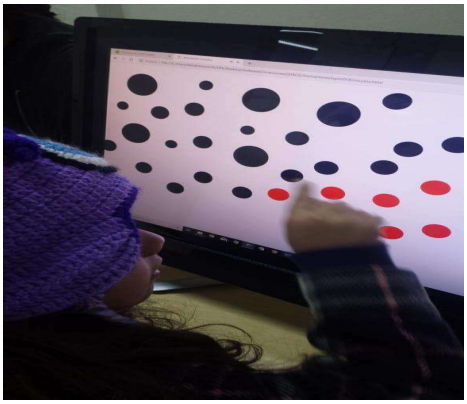

Figura 4.20 Animación Círculos siendo utilizada en el CEIAC.

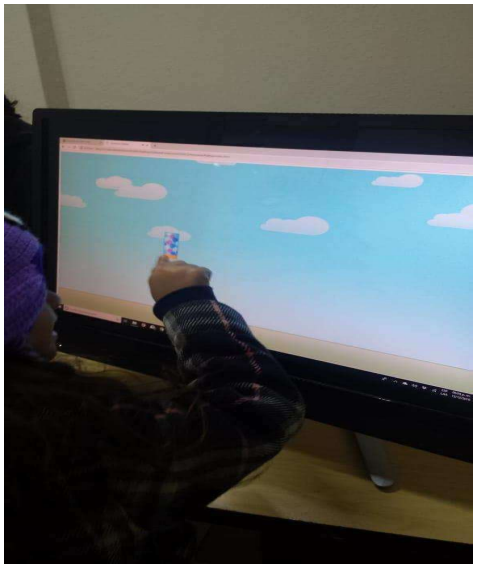

Figura 4.21 Animación Paletas siendo utilizada en el CEIAC.

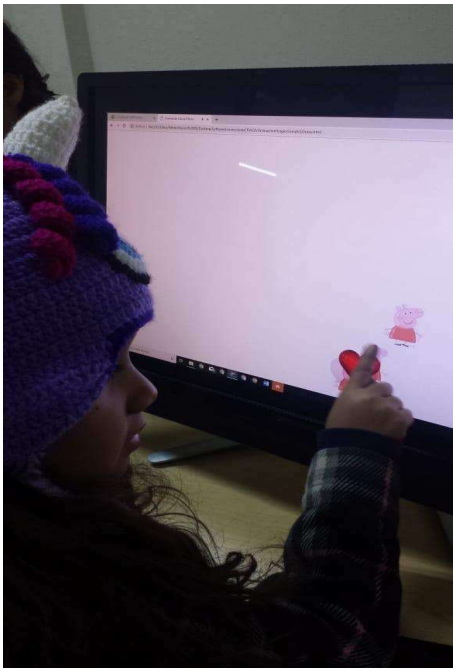

Figura 4.22 Animación Causa y Efecto siendo utilizada en el CEIAC.

V. RESULTADOS Y DISCUSIÓN

En base a los objetivos que se describieron en un principio, se hizo la investigación de todos los temas relacionados con el proyecto, como son los programas que actualmente se utilizan para tratar las deficiencias visuales para niños, además de hacer un estudio amplio de temas relacionados con la deficiencia visual, discapacidad visual, tiflotecnología, también de herramientas y tecnologías utilizadas para el desarrollo del sistema. De todo lo mencionado, se obtuvo un software para la estimulación visual de niños con baja visión, el cual contiene ejercicios muy útiles para los niños que toman la terapia en el Centro de Estudios para Invidentes A.C (CEIAC).

Los objetivos planteados desde un inicio finalmente se cumplieron ya que el software cubre las necesidades que en el Centro de Estudios para Invidentes se solicitaron y específicamente fueron que los ejercicios utilizaran figuras y dibujos animados actuales y que estos fueran llamativos para que el niño se sintiera motivado a la hora de estar tomando la terapia, además que algunos de los ejercicios fueron desarrollados en base a la información que la terapeuta proporcionó, como por ejemplo que se utilizaran algunos estímulos como son los de profundidad, discriminación de colores y rastreo.

VI. CONCLUSIONES.

El software para la estimulación visual de niños con baja visión es una herramienta de trabajo muy útil que utiliza el terapeuta en la terapia de estimulación visual, además de que se logró cumplir el objetivo de tener un software que utilice ejercicios que sean llamativos para que el niño tenga interés a la hora de tomar la terapia, también se cumplió con tarea de tener ejercicios donde se utilizan una variedad de animaciones, donde también se pueda tener un control donde se pueda seleccionar diferentes velocidades con las que se desplieguen las animaciones, trabajar con combinaciones de colores, que ayuden a estimular la vista del niño, además de contar con algunos otros ejercicios donde se pueda ir incrementando la dificultad del ejercicio, esto con el fin de poder utilizarlos con diferentes tipos y diferentes patologías según el grado de deficiencia visual que pueda presentar cada niño que acude a tomar la terapia de estimulación visual.

El desarrollo de este proyecto ayudó a conocer y entender mejor muchos temas con los cuales no se estaba muy relacionados entre los cuales se puede destacar los problemas con los cuales se pueden enfrentar las personas con discapacidad visual, además de poder tener un punto de vista mucho mas empático respecto a las necesidades que las personas con deficiencias visuales presentan, entender que para las personas con capacidades diferentes lo que para algunos puede ser algo tan rutinario y normal, para ellos puede llegar a ase un reto grande de realizar, es muy gratificante saber que este proyecto puede ayudar a mejorar la agudeza visual de los niños que tomen la terapia con este software y que esto impactará positivamente en la vida de los niños, ayudando a que puedan tener una mejor inclusión en la vida diaria y que puedan a tener una mejor integración en todos los ámbitos tanto sociales como de educación, además de que puedan estar lo más actualizados respecto a el uso de las nuevas tecnologías.

Finalmente, conforme el software vaya siendo utilizado, pueden llegar a surgir nuevas ideas, cambios o necesidades respecto al funcionamiento del sistema, por cual es importante llegar a tener una retroalimentación posterior después de haberse utilizado el software.

VII. BIBLIOGRAFÍA

Asociación Doce (2016). Recuperado de https://asociaciondoce.com/2016/11/14/evo-programa-de-estimulacion-visual/

Barraga (1977), Tobin y Chapman (1977:1986), Cantón (1990), De la orden (1990). EVO: Sistema informático de entrenamiento visual para personas deficientes visuales.

Cabrerizo (2000), Arregui (2004). Centro de Investigación, Desarrollo y Aplicaciones Tiflotecnológicas.

Centro de Desarrollo de Tecnologías de Inclusión (s.f). Recursos Tecnológicos y software educativo. Recuperado de http://www.cedeti.cl/recursos-tecnologicos/software-educativo.

Centro de Investigación, Desarrollo y Aplicación Tiflotécnica (2016). Tiflotecnología, Tiflosoftware. Recuperado de: http://cidat.once.es

Centro Para Ciegos y Baja Visión CEIAC. (s.f.) Estimulación visual. Recuperado de: http://www.ceiac.org/

Educación Inclusiva y Discapacidad. (2014). Recuperado de https://cemitic.wordpress.com

Escritorio modalidad educación especial. (s.f). Recuperado de: http://modalidadespecial.educ.ar/

Familias en ruta (s.f.) Recuperado de https://familiasenruta.com/fnrcrianza/educacion/metodo-montessori/

García, C.E. (2012). Guía de atención educativa para estudiantes con discapacidad visual. Instituto de Educación de Aguascalientes

Gauchat, Juan Diego (2012). El gran libro de HTML5, CSS3 y Javascript.

Pérez, P. (2015). Integración, Revista sobre discapacidad visual. Organización Nacional de Ciegos (ONCE). Recuperado de: http://www.once.es/new/servicios-especializados-en-discapacidad-visual

Meralla y Jaritz (1994). Primeros juegos de ordenador empleados como herramientas de estimulación visual.

Netbeans sitio oficial (s.f.). Recuperado de www.netbeans.org

Nuevo, M. (2016). La estimulación visual infantil. Recuperado de: http://www.guiainfantil.com/1454/la-estimulacion-visual-infantil.html

Organización mundial de la salud (OMS) (2014). Ceguera y discapacidad visual. Recuperado de: http://www.who.int/mediacentre/factsheets/fs282/es/

Ramos Salavert, Isidro; Lozano Pérez, María Dolores (2000). Ingeniería del software y bases de datos: tendencias actuales.

Ross (1989), Hammerlund (1994), Arditi y Lei Liu (1996), (Groffman y Solan, 1994). La Medición Optometrica.

Wolfgang von Kempelen (1734-1804), Wolfgang von Kempelen (1734-1804), Roig (2000), Moreno-Montero (2000), Oscar Picht (1871-1945), Ortiz, (2009), Lorente (2009). Historia de la Tiflotecnología en España

ANEXOS

A continuación, se muestra el código que se utilizó para el desarrollo del software:

Código HTML utilizado en la ventana de inicio del sistema y menú con los ejercicios:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Menú principal</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>

    body {
    background: url("niños.jpg") no-repeat center center fixed;
    width: 100%;
    height: auto;
    background-size: cover;
    -moz-background-size: cover;
    -webkit-background-size: cover;
    -o-background-size: cover;
}
.dropbtn {
    background-color: red;
    color: white;
    padding: 25px;
    font-size: 20px;
    border: none;
    cursor: pointer;
}

.dropdown {
    position: relative;
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #ccc;
    font-size: 20px;
    min-width: 160px;
```

```
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}

.dropdown-content a {
    color: darkred;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}

.dropdown-content a:hover {background-color: white}

.dropdown:hover .dropdown-content {
    display: block;
}

.dropdown:hover .dropbtn {
    background-color: darkred;
}
</style>
</head>
<body>

<h2 style="color: mediumblue; font-size: 50px; text-align:
center;">Software para la estimulación visual de niños con baja visión</h2>


<br>
<div class="dropdown">
  <button class="dropbtn">Ejercicios</button>
  <div class="dropdown-content">
    <a href="AnimacionFiguras/index.html">Animación Figuras</a>
    <a href="AnimacionPelotas/seguirPelotas.html">Animación Pelotas</a>
    <a href="AnimacionAtrapar/index.html">Animación Atrapar a pepa!</a>
    <a href="AnimacionImagenSonido2/index.html">Animación causa y
efecto</a>
    <a href="AnimacionPaletas/index.html">Animación paletas</a>
    <a href="AnimacionesSept2018/agua.html">Animación Gotas</a>
    <a href="AnimacionesSept2018/cilindro.html">Animación Cilindro</a>
    <a href="AnimacionesSept2018/circulos.html">Animación Circulos</a>
    <a href="AnimacionesSept2018/lineas.html">Animación Lineas</a>
  </div>
</div>
</body>
</html>
```

Código HTML y JavaScript utilizado en la Animación Figuras.

```
<!DOCTYPE html>

<html>
    <head>
        <title>Animación figuras</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script src="vision.js" type="text/javascript"></script>
    </head>
    <body style="">
        <div id="principal" style="
             float: left;
             width: 81%;
             z-index: -1;">
            <canvas id="lienzo" width="1200" height="800"></canvas>
        </div>
        <div style="
             width: 17%;
             float: right;
             text-align: right;
             border-left: solid;
             padding: 10px;
             height: 100%">
            <hr>
            <h2>Color de fondo</h2>
            <input id="color2" type="color" name="color" value="#ffffff"
                   onchange="
                           principal.style.backgroundColor =
color2.value;">
            <h2>Color de linea</h2>
            <input id="color1" type="color" name="color" value="#FF0000"
                   onchange="cambiarColor(color1.value)">
            <h2>Seleccionar figura</h2>
            <select id="fig" onchange="cambiarFigura(fig.value)">
                <option value="cuadro">Cuadro</option>
                <option value="circulo">Circulo</option>
                <option value="triangulo">Triangulo</option>
                <option value="estrella">Estrella</option>
            </select>

            <br>
            <br>
```

```
            <h3>
                <a href="../menuEjercicios.html">Regresar al
menu<br>principal</a>
            </h3>


        </div>

    </body>
</html>


var canvas;
var c;
var color1, color2;
var colorLinea;
var principal;
var animar;
var inc;
var tipo;

window.onload = function () {
    color1 = document.getElementById("color1");
    color2 = document.getElementById("color2");
    canvas = document.getElementById("lienzo");
    principal = document.getElementById("principal");
    c = canvas.getContext("2d");
    canvas.onclick = function (event) {
        animar = true;
        inc = -inc;
    };
    ciclo();
};
function ciclo() {
    animar = false;
    inc = -1;
    var x = 0;

    c.lineWidth = 10;
    colorLinea = '#ff0000';
    tipo = "cuadro";
    setInterval(function () {
        var mx = canvas.width / 2;
    var my = canvas.height / 2;
        if (animar) {
            c.fillStyle = color2.value;
            c.fillRect(0, 0, canvas.width, canvas.height);
            c.strokeStyle = colorLinea;
```

```
            var x1, y1, t;
            for (var i = 0; i < 20; i++) {
                c.beginPath();
                x1 = mx - (i * 30 + x);
                y1 = my - (i * 30 + x);
                t = (i * 30 + x) * 2;
                switch (tipo) {
                    case "cuadro":
                        c.strokeRect(x1, y1, t, t);
                        break;
                    case "circulo":
                        c.arc(mx, my, Math.abs(t / 4 * 3), 0, 2 * Math.PI);
                        break;
                    case "triangulo":
                        dibujarTriangulo(mx, my, t * 2, t * 2);
                        break;
                    case "estrella":
                        dibujarEstrella(mx, my, 5, t * 2, t);
                        break;
                }
                c.stroke();
            }
            x += inc;
            if (x >= 30) {
                x = 0;
            }
        }
    }, 20);
}
function dibujarTriangulo(x, y, tx, ty) {
    c.moveTo(x - tx / 2, y + ty / 2);
    c.lineTo(x + tx / 2, y + ty / 2);
    c.lineTo(x, y - ty);
    c.closePath();
}
function dibujarEstrella(cx, cy, spikes, outerRadius, innerRadius) {
    var rot = Math.PI / 2 * 3;
    var x = cx;
    var y = cy;
    var step = Math.PI / spikes;

    c.beginPath();
    c.moveTo(cx, cy - outerRadius);
    for (i = 0; i < spikes; i++) {
        x = cx + Math.cos(rot) * outerRadius;
        y = cy + Math.sin(rot) * outerRadius;
        c.lineTo(x, y);
        rot += step;
```

```
        x = cx + Math.cos(rot) * innerRadius;
        y = cy + Math.sin(rot) * innerRadius;
        c.lineTo(x, y);
        rot += step;
    }
    c.lineTo(cx, cy - outerRadius);
    c.closePath();
}
function limite(v, menor, mayor) {
    if (v < menor) {
        return menor;
    }
    if (v > mayor) {
        return mayor;
    }
    return v;
}
function cambiarFigura(f) {
    tipo = f;
    inc = 1;
}
function cambiarColor(c) {
    colorLinea = c;
}
```

Código HTML y JavaScript utilizado en la Animación Pelotas.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset=utf-8 />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <title>Animación pelotas</title>
        <style>
            canvas {
                border: 2px solid #ff5722;
            }

            img {
                width: 60px;
                height: 60px;
            }
            div {width: 100px;
```

```
                height: 100px;
                background-color: "#990000";}
            </style>
            <script>
                var canvas, ctx, width, height;
                var ballArray = [];

                var colorpelota1 = "#990000";
                var numpelotas1 = 10;

                function init() {
                    canvas = document.querySelector("#myCanvas");
                    ctx = canvas.getContext('2d');
                    width = canvas.width;
                    height = canvas.height;

                    // Escuchador para color de pelota a seguir
                    var colorpelota =
document.getElementById("colorpelota");
                    colorpelota.addEventListener('change', cambiarcolor,
false);

                    // Escuchador para número de pelotas
                    var numpelotas = document.getElementById("numpelotas");
                    numpelotas.addEventListener('change',
cambiarnumpelotas, false);

                    // Change this number to get more balls
                    createBalls(numpelotas1);

                    requestAnimationFrame(mainLoop);
                }

                function createBalls(numberOfBalls) {
                    ballArray = [];
                    for (var i = 0; i < numberOfBalls; i++) {

                        // Create a ball with random position and speed
                        var ball = new Ball(width * Math.random(),
                                height * Math.random(),
                                (10 * Math.random()) - 5,
                                (10 * Math.random()) - 5,
                                40,
                                i); // radius, change if ou like.

                        // Add it to the array
                        ballArray[i] = ball;
                    }
```

```
}

function mainLoop() {
    // vasClear the can
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    // uncomment for blur effect, comment previous line
    //ctx.fillStyle = "rgba(0, 240, 240, 0.2)";
    //ctx.fillRect (0, 0, width, height);
    //ctx.fillStyle='black';

    // For each ball in the array
    for (var i = 0; i < ballArray.length; i++) {
        var balle = ballArray[i];

        // 1) Move the ball
        balle.move();

        // 2) collision test with walls
        collisionTestWithWalls(balle);

        // 3) draw the ball
        balle.draw();
    }

    collisionTestBetweenBalls();

    // Ask for new animation frame
    window.requestAnimationFrame(mainLoop);
}

function collisionTestWithWalls(ball) {
    if (ball.x < ball.rayon) {
        ball.x = ball.rayon;
        ball.vx *= -1;
    }
    if (ball.x > width - (ball.rayon)) {
        ball.x = width - (ball.rayon);
        ball.vx *= -1;
    }
    if (ball.y < ball.rayon) {
        ball.y = ball.rayon;
        ball.vy *= -1;
    }
    if (ball.y > height - (ball.rayon)) {
        ball.y = height - (ball.rayon);
        ball.vy *= -1;
    }
```

```
                }

                function collisionTestBetweenBalls() {
                    var balls = ballArray;

                    for (var i = 0; i < ballArray.length; i++) {
                        for (var j = i + 1; j < ballArray.length; j++) {
                            var dx = balls[j].x - balls[i].x;
                            var dy = balls[j].y - balls[i].y;

                            var dist = Math.sqrt(dx * dx + dy * dy);
                            if (dist < (balls[j].rayon + balls[i].rayon)) {
                                // balls have contact so push back...
                                var normalX = dx / dist;
                                var normalY = dy / dist;
                                var middleX = (balls[i].x + balls[j].x) /
2;
                                var middleY = (balls[i].y + balls[j].y) /
2;

                                balls[i].x = middleX - normalX *
balls[i].rayon;
                                balls[i].y = middleY - normalY *
balls[i].rayon;
                                balls[j].x = middleX + normalX *
balls[j].rayon;
                                balls[j].y = middleY + normalY *
balls[j].rayon;

                                var dVector = (balls[i].vx - balls[j].vx) *
normalX;
                                dVector += (balls[i].vy - balls[j].vy) *
normalY;
                                var dvx = dVector * normalX;
                                var dvy = dVector * normalY;

                                balls[i].vx -= dvx;
                                balls[i].vy -= dvy;
                                balls[j].vx += dvx;
                                balls[j].vy += dvy;
                            }
                        }
                    }
                }

                function Ball(x, y, vx, vy, diameter, id) {
                    this.x = x;
                    this.y = y;
```

```
                    this.vx = vx;
                    this.vy = vy;
                    this.rayon = diameter / 2;
                    this.id = id;

                    this.draw = function () {
                        if (this.id === 2) {
                            ctx.fillStyle = colorpelota1;
                        } else {
                            ctx.fillStyle = "#000099";
                        }
                        ctx.beginPath();
                        ctx.arc(this.x, this.y, this.rayon, 0, 2 *
Math.PI);

                        ctx.fill();

                    };

                    this.move = function () {

                        this.x += this.vx;
                        this.y += this.vy;
                    };

                }

                function cambiarcolor() {
                    // alert("cambiar color a:");
                    colorpelota1 =
document.getElementById("colorpelota").value;
                    document.getElementById('colorpelota2').value =
colorpelota1;
                }

                function cambiarnumpelotas() {
                    numpelotas1 =
document.getElementById("numpelotas").value;
                    if (numpelotas1 < 3 || numpelotas1 > 30 || numpelotas1
=== "") {
                        numpelotas1 = 10;
                    }
                    //alert("Num pelotas: " + numpelotas1);
                    createBalls(numpelotas1);


                }
```

```
            </script>
        </head>
        <body onload="init();" style=" background-color: lightskyblue">
        <h2 style="color: crimson">Ejercicio:    Seguir una pelota de
color</h2>
        <table>
            <tr>
                <td style="color: blue; " >
                    Elige un color:     
                    <input type="color" id="colorpelota"
                           name="colorpelota">
                </td>
                <td style="color: blue">
                    Elige el número de pelotas (3..30):
                      
                    <input type="number" min="3" max="30"
                           id="numpelotas" name="numpelotas"
                           value="10">

                </td>
            </tr>
            <tr>
                <td style="color: blue">
                    Sigue pelota de color:
                      
                    <input type="color" id="colorpelota2"
name="colorpelota2"
                           value="#990000" disabled>
                </td>
                <td>

                </td>
            </tr>
        </table>

        <canvas id="myCanvas" width="1000" height="500">
        </canvas>

        <div>
            <h3>
                <a href="../menuEjercicios.html">Regresar al menu
principal</a>
            </h3>
        </div>
    </body>
</html>
```

Código HTML y JavaScript utilizado en la Animación Atrapar a Pepa!

```
<!DOCTYPE html>
<html>
    <head>
        <title>Animación Atrapar</title>

        <meta charset='utf-8'>


        <!-- freedom from browser prefixes, yay! -->
        <!-- javascript for the slides -->
        <script type="text/javascript" src='prescommon/slides.js'></script>

        <!-- our Box2D code -->
        <script type="text/javascript"
src="canvas_libraries/canvas_utils.js"></script>

        <!-- canvas slides code -->
        <script type="text/javascript"
src="canvas_libraries/slide_transforms.js"></script>

    </head>

    <style>
        /* Your individual styles here, or just use inline styles if that’s
           what you want. */


        .animcontainer
        {
            position: relative;
            width: 100%;
            height: 270px;
        }

        table.table-noborders tr td
        {
            border: none;
            padding: 0;
            margin: 0;
        }

        table.table-noborders
        {
```

```
            margin:0;
        }

        .scene3d
        {
            perspective: 1000px;
            width: 600px;
            height: 340px;
            margin-left: auto;
            margin-right: auto;
        }
        .object3d
        {
            position: absolute;
            width: inherit;
            height: inherit;
            /*top: 20px;*/
            transform-style: preserve-3d;
        }
        .face3d
        {
            position: absolute;
            left: 165px;
            top: 15px;
        }
        pre.big
        {
            font-size: 0.825em;
        }

        /* CSS to be used with "spacelaunch" script to turn the
           current slide into a luanchpad and jump into 3D space. */

        body.spacelaunch
        {
            background: rgb(0, 0, 0);
        }
        .modellaunch .slides > article.modellaunch-central
        {
            transform: rotateX(80deg) translateY(50px);
            transform-origin: 50% 100%;
            transition: 0.75s;
            transform-style: preserve-3d;
        }
        .spacelaunch .slides > article.spacelaunch-central
        {
            transform: rotateX(100deg) translateY(1300px) translateZ(-
600px);
```

```
            transform-origin: 50% 100%;
            transition: 2s;
        }
        .spacelaunch .slides > article.spacelaunch-right
        {
            transform: translate(1020px) translateZ(500px);
        }
        .spacelaunch .slides > article.spacelaunch-left
        {
            transform: translate(-1020px) translateZ(500px);
        }
    </style>

    <body style="display: none; transition: 2s" id="canvas3D">

        <section class='slides layout-regular template-default'
style="perspective: 1000px" lang="fr">

            <!-- Your slides (<article>s) go here. Delete or comment out
the
                 slides below. -->

            <!-- ------------------------------------------------------------
--------------------- -->
            <!-- HTML canvas: geometric transforms -->

            <style type="text/css">
                #html-canvas-transforms table
                { border: none; margin-bottom: 10px }
                #html-canvas-transforms table tr td
                { border: none; text-align: center;}
                #html-canvas-transforms table tr td pre
                { text-align: left; margin-bottom: 15px;}
            </style>

            <article onslideenter="slide_canvas_transforms()" id="html-
canvas-transforms" style="overflow: visible;">
                <canvas id="canvas-transforms" style="width: 100%; height:
150%; position: absolute; bottom: 0px; left: 0px;"></canvas>

                <translation lang="en">
                    Canvas: geometric transforms
                    --self--
                    draw here
                    Translation
                    --self--
                    draw here
                    Rotation
```

```
                    --self--
                    draw here
                    Scale
                    use 
                    --self--
                    and 
                    --self--
                    to handle drawing ctx. state
                </translation>
                <translation lang="pt">
                    Canvas: transformações geométricas
                    --self--
                    desenhe aqui
                    Mudar de posição
                    --self--
                    desenhe aqui
                    Rotação
                    --self--
                    desenhe aqui
                    Aumentar/Diminuir
                    use 
                    --self--
                    e 
                    --self--
                    para manipular o estado do ctx.
                </translation>

            </article>

            <!-- ------------------------------------------------------------
--------------------- -->
        </section>

        <br>
        <div>

                <a href="../menuEjercicios.html"></a>

        </div>
    </body>
</html>


function modelshow(el, modelURL)
{
     if (animation !== null)
          return;
```

```
    el.classList.add('modellaunch-central');
    document.body.classList.add('modellaunch');
    el.onclick = function()
    {
        modelhide(el, modelURL);
    };


    // dimensions
    var canvascontainer = document.getElementById('modelcontainer');
    var width = canvascontainer.clientWidth;
    var height = canvascontainer.clientHeight-2; // unfathomable bug: why
do I get a scroll bar with the full height ???

    // renderer
    var renderer = new THREE.WebGLRenderer ({alpha:true, antialias:
true});
    renderer.setSize(width, height);

    // glue to HTML element
    canvascontainer.insertBefore(renderer.domElement,canvascontainer.first
Child);
    renderer.setClearColor(0x000000, 0);

    // camera
    var camera = new THREE.PerspectiveCamera(35, width / height, 1, 3000);
    camera.position.x = 0;
    camera.position.y = 40;
    camera.position.z = 110;
    camera.rotation.x = -Math.PI/2/10;

    // scene
    var scene = new THREE.Scene();

    // lights
    var light1 = new THREE.DirectionalLight(0xffffff, 1);
    light1.position.set(1, 1, 0.3);
    scene.add(light1);

    var light2 = new THREE.DirectionalLight(0xffffff, 1);
    light2.position.set(-1, 1, -0.3);
    scene.add(light2);

    var light3 = new THREE.DirectionalLight(0xffffff, 0.4);
    light3.position.set(0, 0, 1);
    scene.add(light3);

    // launch animation
```

```
     animation = new Object();
     animation.renderer = renderer;
     animation.scene    = scene;
     animation.camera   = camera;
     animation.controls = null;
     animation.clock    = null;
     animation.gettingclose_event = null;
     window.requestAnimationFrame(animate);

     var rot0 = new THREE.Euler(0,0,0);
     var vec111 = new THREE.Vector3(1,1,1);

     var sequencer = new Object();
     sequencer.todo = new Object();
     sequencer.begin = 0;
     sequencer.end = 0;
     sequencer.add = function(f) {sequencer.todo[sequencer.end++] = f;};
     sequencer.go = function() {if (sequencer.begin < sequencer.end)
sequencer.todo[sequencer.begin++]();};

     loadModel(scene, modelURL, new THREE.Vector3(0,0,0), rot0, vec111,
"androidmodel-gingerbread", sequencer);

     window.setTimeout(function(){sequencer.go();}, 1000);
}

function modelhide(el, modelURL)
{
     var canvascontainer = document.getElementById('modelcontainer');
     el.classList.remove('modellaunch-central');
    document.body.classList.remove('modellaunch');
     el.onclick = function() {modelshow(el, modelURL);};

     // kill the 3D canvas too
     window.setTimeout( function() {
     if (canvascontainer !== null && canvascontainer.firstChild.tagName ==
'CANVAS')
          canvascontainer.removeChild(canvascontainer.firstChild);
     if (animation !== null && animation.controls !== null)
          animation.controls.removeEventListeners();
     animation = null;
     }, 100);
}

function spaceland()
{
     var el     = document.querySelector('.current');
     var elNext = document.querySelector('.next');
```

```
     var elPrev = document.querySelector('.past');
     var body   = document.querySelector('body');

     if (el !== null)
          el.classList.remove('spacelaunch-central');
     if (elNext !== null)
          elNext.classList.remove('spacelaunch-right');
     if (elPrev !== null)
          elPrev.classList.remove('spacelaunch-left');
     if (body !== null)
          body.classList.remove('spacelaunch');

     // kill the 3D canvas too
     window.setTimeout( function() {
     if (body !== null && body.firstChild.tagName == 'CANVAS')
          body.removeChild(body.firstChild);
     if (animation !== null)
          animation.controls.removeEventListeners();
     animation = null;
     }, 100);
}

function spacelaunch()
{
     var el     = document.querySelector('.current');
     var elNext = document.querySelector('.next');
     var elPrev = document.querySelector('.past');
     var body   = document.querySelector('body');

     el.classList.add('spacelaunch-central');
     elNext.classList.add('spacelaunch-right');
     elPrev.classList.add('spacelaunch-left');
     body.classList.add('spacelaunch');

     // dimensions
     var canvascontainer = document.getElementById('canvas3D');
     var width = canvascontainer.clientWidth;
     var height = canvascontainer.clientHeight-4; // unfathomable bug: why
do I get a scroll bar with the full height ???

     // renderer
     var renderer = new THREE.WebGLRenderer ({antialias: true});
     renderer.setSize(width, height);

     // glue to HTML element
     canvascontainer.insertBefore(renderer.domElement,canvascontainer.first
Child);
     renderer.setClearColor(0x000000, 0);
```

```
    // camera
    var camera = new THREE.PerspectiveCamera(35, width / height, 1, 3000);
    camera.position.x = 0;
    camera.position.y = 360;
    camera.position.z = 1800;
    camera.rotation.x = -Math.PI/2/10;

    // scene
    var scene = new THREE.Scene();

    //var controls = new THREE.MyFlyControls(camera, renderer.domElement);
    var controls = new THREE.MyFlyControls(camera, document.body); //
unfathomable bug: if the canvas is passed here then events are never
received
    controls.movementSpeed = 100;
    controls.rollSpeed = 0.5;
    //controls.autoForward = true;

    // READY >

    // lights
    var light1 = new THREE.DirectionalLight(0xffffff, 1);
    light1.position.set(1, 1, 0.3);
    scene.add(light1);

    var light2 = new THREE.DirectionalLight(0xffffff, 1);
    light2.position.set(-1, 1, -0.3);
    scene.add(light2);

    var light3 = new THREE.DirectionalLight(0xffffff, 0.4);
    light3.position.set(0, 0, 1);
    scene.add(light3);

    // ground plane
    var groundTex = THREE.ImageUtils.loadTexture('../models/ground.png');
    groundTex.wrapS = groundTex.wrapT = THREE.RepeatWrapping;
    groundTex.repeat.set(50, 50);
    var groundGeo = new THREE.BoxGeometry(3000, 3000, 10, 1, 1, 1);
    var groundMat = new THREE.MeshBasicMaterial({map: groundTex,
emissive:0x444444});
    var ground    = new THREE.Mesh(groundGeo, groundMat);
    ground.rotation.x = Math.PI/2;
    ground.rotation.z = 1;
    ground.translateZ(15);
    scene.add(ground);

    // launch animation
```

```
    animation = new Object();
    animation.renderer = renderer;
    animation.scene    = scene;
    animation.camera   = camera;
    animation.controls = controls;
    animation.clock    = new THREE.Clock();
    animation.gettingclose_event = spacelaunch_close;
    animation.gettingclose_event_fired = false;
    window.requestAnimationFrame(animate);

    var rot0 = new THREE.Euler(0,0,0);
    var vec111 = new THREE.Vector3(1,1,1);

    var sequencer = new Object();
    sequencer.todo = new Object();
    sequencer.begin = 0;
    sequencer.end = 0;
    sequencer.add = function(f) {sequencer.todo[sequencer.end++] = f;};
    sequencer.go = function() {if (sequencer.begin < sequencer.end)
sequencer.todo[sequencer.begin++]();};
    animation.sequencer = sequencer;
    animation.sequencer_fired = false;

    loadModel(scene, "../models/Cupcake.dae",          new THREE.Vector3(-
200,0,0), rot0, vec111, "androidmodel-cupcake", sequencer);
    loadModel(scene, "../models/Donut.dae",            new THREE.Vector3(-
150,0,0), rot0, vec111, "androidmodel-donut", sequencer);
    loadModel(scene, "../models/Eclair.dae",           new THREE.Vector3(-
100,0,0), rot0, vec111, "androidmodel-eclair", sequencer);
    loadModel(scene, "../models/Froyo.dae",            new THREE.Vector3(-
50,0,0),  rot0, vec111, "androidmodel-froyo", sequencer);
    loadModel(scene, "../models/Gingerbread.dae",      new
THREE.Vector3(0,0,0),    rot0, vec111, "androidmodel-gingerbread",
sequencer);
    loadModel(scene, "../models/Honeycomb.dae",        new
THREE.Vector3(50,0,0),   rot0, vec111, "androidmodel-honeycomb",
sequencer);
    loadModel(scene, "../models/IceCreamSandwich.dae", new
THREE.Vector3(100,0,0),  rot0, vec111, "androidmodel-icecreamsandwich",
sequencer);
    loadModel(scene, "../models/JellyBean.dae",        new
THREE.Vector3(150,0,0),  rot0, vec111, "androidmodel-jellybean",
sequencer);
    loadModel(scene, "../models/KitKat.dae",           new
THREE.Vector3(200,0,0),  rot0, vec111, "androidmodel-kitkat", sequencer);
}

function spacelaunch_close()
```

```
{
    // launch loading sequence when you get sufficiently close to the
models
    animation.sequencer.go();
}

function loadModel(scene, url, position, rotation, scale, name, sequencer)
{
    // instantiate a loader
    var loader = new THREE.ColladaLoader();
    loader.options.convertUpAxis= true;
    loader.options.upAxis = 'Y';

    var loadheight = 46;

    // create a loding shape (kind of 3D progres bar)
    var loadingGeometry = new THREE.CylinderGeometry(20, 20, loadheight,
20, 10, false);
    var loadingMaterial = new THREE.MeshBasicMaterial({color: 0x8888ff,
wireframe: true, wireframeLinewidth: 1});
    var loadingShape    = new THREE.Mesh(loadingGeometry,
loadingMaterial);

    var loadedGeometry  = new THREE.CylinderGeometry(15, 15, loadheight,
20, 10, false);
    var loadedMaterial  = new THREE.MeshLambertMaterial({color: 0xffffff,
emissive: 0x444444});
    var loadedShape     = new THREE.Mesh(loadedGeometry, loadedMaterial);

    loadingShape.name = "loading-" + name;
    loadingShape.translateX(position.x);
    loadingShape.translateY(position.y + loadheight/2);
    loadingShape.translateZ(position.z);
    loadingShape.rotation.copy(rotation);

    loadedShape.scale.set(1, 0.01, 1);
    loadedShape.name = "loaded-" + name;
    loadedShape.translateX(position.x);
    loadedShape.translateY(position.y + 0.01*loadheight/2);
    loadedShape.translateZ(position.z);
    loadedShape.rotation.copy(rotation);

    // make the loading shape appear
    scene.add(loadingShape);
    scene.add(loadedShape);

    // add the actual loading task into a sequential queue
    sequencer.add( function() {
```

```
          loader.load(url,
                    function(collada)
                    {
                         console.log("MARTIN: finished loading model " +
name);
                         var model = collada.scene;
                         model.position.copy(position);
                         model.rotation.copy(rotation);
                         model.scale.copy(scale);
                         model.name = name;
                         setProgress3D(scene, model, loadingShape,
loadedShape, loadheight, 100);

                         window.setTimeout(function(){sequencer.go();}, 300);
                    },
                    function(progress)
                    {
                         var percent = 0;
                         if (progress.lengthComputable)
                              percent = progress.loaded / progress.total*100;
                         else
                              percent = progress.loaded / (1024*1024) * 100;
                         console.log("MARTIN: progress loading model " + name
+ ":" + percent + "%");
                         setProgress3D(scene, null, loadingShape,
loadedShape, loadheight, percent);
                    }
          );}
     );
}

function setProgress3D(scene, model, waitShape, progressShape,
progressShapeMaxHeight, percent)
{
     var f = percent/100;
     if (f>1)
          f = 1;
     var t0 = new Date().getTime();
     var f0 = progressShape.scale.y;
     if (f0 == f)
          return;
     progressShape.continuousUpdate = function(t)
     {
          var duration = 300;
          var intermediateScale = f;
          if (t-t0 <= duration)
               intermediateScale = (t-t0)/duration*f + (1-(t-
t0)/duration)*f0;
```

```
          progressShape.translateY((intermediateScale-
progressShape.scale.y)*progressShapeMaxHeight/2);
          progressShape.scale.set(1, intermediateScale, 1);

          // if animation time exhausted
          // undefine the function and if the loading is finished, also swap
in the real model
          if (t-t0 > duration)
          {
               if (f == 1)
               {
                    scene.remove(waitShape);
                    scene.remove(progressShape);
                    if (model !== null)
                         scene.add(model);
               }
               delete this.continuousUpdate;
          }
     };
}

function animate()
{
     var t = new Date().getTime();

     // if no animation, return and don't relaunch animation loop
     if (animation == null)
          return;

     var renderer = animation.renderer;
     var scene    = animation.scene;
     var camera   = animation.camera;
     var controls = animation.controls;
     var clock    = animation.clock;

     var objectsToUpdate = new Object();
     var nbObjectsToUpdate = 0;

     scene.traverse(function(object)
     {
          // models: slow rotation
          if (object instanceof THREE.Object3D &&
object.name.indexOf("androidmodel") >= 0)
          {
               object.rotation.y = t/1000;
          }
```

```
          // loading shapes: fast rotation
          if (object instanceof THREE.Object3D &&
object.name.indexOf("loading") >= 0)
          {
             object.rotation.y = t/500;
          }

          // loading progress shapes: continuously update height
          if (object instanceof THREE.Object3D &&
object.name.indexOf("loaded") >= 0)
          {
               if (object.continuousUpdate !== undefined)
                    objectsToUpdate[nbObjectsToUpdate++] = object;
          }
     });

     // the progress shape animation also switches in the final loading
shape
     // when it finishes so these calls must be made outside of the scene
traversal loop
     for (var i=nbObjectsToUpdate-1; i>=0; i--)
     {
          objectsToUpdate[i].continuousUpdate(t);
          delete objectsToUpdate[i];
     }

     if (clock !== null && controls !== null)
     {
          var delta = clock.getDelta();
          controls.update(delta);
     }
     renderer.render(scene, camera);

     // launch an event if the camera gets closer than 900px
     if (animation.gettingclose_event !== null &&
!animation.gettingclose_event_fired && camera.position.lengthSq() <
900*900)
     {
          animation.gettingclose_event();
          animation.gettingclose_event_fired = true;
     }

     // keep going
     window.requestAnimationFrame(animate);
}

function move(id)
{
```

```
     var nod = document.getElementById(id);
     if (nod != null)
     {
          if (!nod.classList.contains('begin') &&
!nod.classList.contains('end'))
               nod.classList.add('end');
          else
          {
               nod.classList.toggle('begin');
               nod.classList.toggle('end');
          }
     }
}

function step(id, classroot, max)
{
     var el = document.getElementById(id);
     for (var i=1; i<=max; i++)
     {
          if (el.classList.contains(classroot + i))
          {
               el.classList.remove(classroot + i);
               if (i+1<=max)
                    el.classList.add(classroot + (i+1));
               else
                    el.classList.add(classroot + 1);
               break;
          }
     }
}

function stepSVGanimation(id, stepCount, stepTime)
{
       if (window['animstate-'+id] === undefined || window['animstate-'+id]
== 0 || window['animstate-'+id] > stepCount)
       {
            document.getElementById(id).pauseAnimations();
            document.getElementById(id).setCurrentTime(0);
            window['animstate-'+id] = 1;
       }
       else if (window['animstate-'+id] <= stepCount)
       {
            document.getElementById(id).unpauseAnimations();
            window['animstate-'+id] += 1;
            if (window['animstate-'+id] <= stepCount)
                 setTimeout(function
timeout(){document.getElementById(id).pauseAnimations();}, stepTime);
       }
```

```
}

function toggleSVGanimation(id)
{
       if (window['binaryanimstate-'+id] === undefined ||
window['binaryanimstate-'+id] == 0)
       {
            document.getElementById(id).pauseAnimations();
            window['binaryanimstate-'+id] = 1;
       }
       else if (window['binaryanimstate-'+id] >= 1)
       {
            document.getElementById(id).unpauseAnimations();
            window['binaryanimstate-'+id] = 0;
       }
}

function stepSplineSVGanimation()
{
     var stringValues = ["0 0 1 1", "0 0.6 0.4 1", "0.35 0 0.65 1"];
     var splineline1dValues = ["M 100,100 L 100,100", "M 100,100 L 100,40",
"M 100,100 L 100,65"];
     var splineline2dValues = ["M 0,0 L 0,0", "M 0,0 L 60,0", "M 0,0 L
0,35"];
     var splinecircle1cxValues = [100, 100, 100];
     var splinecircle1cyValues = [100,  40,  65];
     var splinecircle2cxValues = [  0,  60,   0];
     var splinecircle2cyValues = [  0,   0,  35];
     var splinecurve1dValue = ["M 0,0 L 100,100", "M 0,0 C 60,0 100,40
100,100", "M 0,0 C 0,35 100,65 100,100"];

     if (window['splineanimstate'] === undefined ||
window['splineanimstate'] == 3)
          window['splineanimstate'] = 0;
     else
          window['splineanimstate'] += 1;

     var i = window['splineanimstate'];

     document.getElementById('wavesurf4').setAttributeNS(null,"keySplines",
stringValues[i]+"; "+stringValues[i]);
     document.getElementById('splinecnt1').firstChild.innerHTML =
stringValues[i]+"; "+stringValues[i];
     document.getElementById('splinecnt2').innerHTML = stringValues[i];
     document.getElementById('splinecircle1').setAttributeNS(null,"cx",
splinecircle1cxValues[i]);
     document.getElementById('splinecircle1').setAttributeNS(null,"cy",
splinecircle1cyValues[i]);
```

```
    document.getElementById('splinecircle2').setAttributeNS(null,"cx",
splinecircle2cxValues[i]);
    document.getElementById('splinecircle2').setAttributeNS(null,"cy",
splinecircle2cyValues[i]);
    document.getElementById('splineline1').setAttributeNS(null,"d",
splineline1dValues[i]);
    document.getElementById('splineline2').setAttributeNS(null,"d",
splineline2dValues[i]);
    document.getElementById('splinecurve1').setAttributeNS(null,"d",
splinecurve1dValue[i]);
}
```

Código HTML y JavaScript utilizado en la Animación Causa y efecto.

```
!DOCTYPE html>

<html>
    <head>
        <title>Animación Causa/Efecto</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script src="vision.js"></script>
    </head>
    <body style="">
        <div style="width: 100%;
             height: 100%;">
            <div id="principal" style="float: left;
                 width: 80%;
                 border-right: solid;">
                <canvas id="lienzo" onmousedown="mouse(event)"></canvas>
            </div>
            <div style="
                 width: 17%;
                 float: right;
                 text-align: right;
                 padding: 10px;
                 height: 100%">
                <button onclick="animar = true;
                        if (pj.length === 0) {
                            reiniciar();
                        }">Iniciar</button>
                <button onclick="animar = false;">Detener</button>
                <hr>
                <h2>Imagen</h2>
```

```
                <select id="selimg"
                        onchange="reiniciar()">
                    <option value="0">Pepa Pig</option>
                    <option value="1">Minion </option>
                </select>
                <hr>
                <h2>Velocidad</h2>
                <form action="">
                    <input id="vel1" type="radio" name="vel" value="Lento">
Lento<br>
                    <input id="vel2" type="radio" name="vel" value="Medio"
checked> Medio<br>
                    <input id="vel3" type="radio" name="vel"
value="Rapido"> Rapido<br>
                </form>
                <hr>
                <h2>Color de fondo</h2>
                <input id="color" type="color" name="color" value="#ffffff"
                       onchange="
                               console.log('si');
                               principal.style.backgroundColor =
color.value;">
                <br>
                <br>
                <h3>
                    <a href="../menuEjercicios.html">Regresar al menu
principal</a>
                </h3>
            </div>
        </div>
    </body>
</html>


var canvas;
var c;
var animar;
var color;
var vel1, vel2, vel3;
var selimg;
var pj;
var principal;
var efecto;
var efectoImg;
window.onload = function () {
    animar = true;
    selimg = document.getElementById("selimg");
    vel1 = document.getElementById("vel1");
```

```
    vel2 = document.getElementById("vel2");
    vel3 = document.getElementById("vel3");
    color = document.getElementById("color");
    canvas = document.getElementById("lienzo");
    principal = document.getElementById("principal");
    c = canvas.getContext("2d");
    efectoImg = new Textura("anim_corazon2.png");
    inicializar("pepa_baile.png",
            new Punto2D(5, 1), new Animacion([0, 1, 2, 3, 4],
            100));
    ciclo();
};
function inicializar(idimagen, totalcuadros, anim) {
    var textura = new Textura(idimagen);
    //var nivel = new Textura("nivel.jpg");
    textura.getImagen().onload = function () {
        pj = new Array();
        efecto = new Array();
        var s;
        for (var i = 0; i < 5; i++) {
            s = new SpriteEx(new Sprite(textura,
                    totalcuadros,
                    anim));
            s.getSprite().setPos(new Punto2D(rnd(0, canvas.width - 100),
                    rnd(0, canvas.height - 128)));
            s.getSprite().setTam(new Punto2D(150, 192));
            pj.push(s);
        }
    };
}
function ciclo() {
    var ti = new Date().getTime();
    setInterval(function () {
        principal.clientheight = window.innerHeight - 10;
        canvas.width = principal.clientWidth;
        canvas.height = principal.clientheight;
        c.fillStyle = color.value;
        c.fillRect(0, 0, canvas.width, canvas.height);
        //c.drawImage(nivel.getImagen(), 0, 0, canvas.width,
canvas.height);
        for (var i = 0; i < pj.length; i++) {
            pj[i].getSprite().dibujar(c);
        }
        if (animar) {
            if (new Date().getTime() - ti > velocidad()) {
                for (var i = 0; i < pj.length; i++) {
                    pj[i].actPos();
                }
```

```
                ti = new Date().getTime();
            }
            for (var i = 0; i < efecto.length; i++) {
                efecto[i].dibujar(c);
            }
        }
    }, 1);
}
function reiniciar() {
    switch (selimg.options[selimg.selectedIndex].value) {
        case '0':
            inicializar('pepa_baile.png',
                    new Punto2D(5, 1), new Animacion([0, 1, 2, 3, 4],
                    100));
            break;
        case '1':
            inicializar('minion.png',
                    new Punto2D(25, 14),
                    new Animacion([0, 1, 2, 3, 4],
                            100));
            break;
    }
}
function mouse(e) {
    var x = e.clientX;
    var y = e.clientY;
    for (var i = 0; i < pj.length; i++) {
        var pos = pj[i].getSprite().getPos();
        var tam = pj[i].getSprite().getTam();
        if (x >= pos.getX() && x <= pos.getX() + tam.getX() &&
                y >= pos.getY() && y <= pos.getY() + tam.getY()) {
            pj.splice(i, 1);
            var sp = new Sprite(efectoImg, new Punto2D(5, 1),
                    new Animacion([0, 1, 2, 3, 4],
                            300));
            //sp.setTam(sp.getTam().getX() / 2, sp.getTam().getY() / 2);
            sp.setPos(new Punto2D(x - sp.getTam().getX() / 2,
                    y - sp.getTam().getY() / 2));
            sp.setLoop(false);
            efecto.push(sp);
            var audio = new Audio('magical_3.ogg');
            audio.play();
            if (pj.length === 0) {
                //alert("Ganaste!");
            }
        }
    }
}
```

```
velocidad = function () {
    if (vel1.checked) {
        return 50;
    }
    if (vel2.checked) {
        return 25;
    }
    if (vel3.checked) {
        return 10;
    }
};
function SpriteEx(sprite) {
    var ix = rnd(10, 30) / 5;
    var iy = rnd(10, 30) / 5;
    this.actPos = function () {
        sprite.setPos(new Punto2D(sprite.getPos().getX() + ix,
                sprite.getPos().getY() + iy));
        if (sprite.getPos().getX() >= (canvas.width - 130) ||
                sprite.getPos().getX() <= 0) {
            ix = -ix;
        }
        if (sprite.getPos().getY() >= (canvas.height - 180) ||
                sprite.getPos().getY() <= 0) {
            iy = -iy;
        }
    };

    this.getSprite = function () {
        return sprite;
    };
}

function Sprite(img, numCuadros, animacion) {
    var pos = new Punto2D(0, 0);
    var loopanim = true;
    var visible = true;
    var tamCuadro = new Punto2D(img.getImagen().width / numCuadros.getX(),
            img.getImagen().height / numCuadros.getY());
    var tam = new Punto2D(tamCuadro.getX(), tamCuadro.getY());
    var indice = 0;
    var ti = new Date().getTime();
    var cuadros = new Array();
    for (var i = 0; i < numCuadros.getY(); i++) {
        for (var j = 0; j < numCuadros.getX(); j++) {
            cuadros.push(new Punto2D(j * tamCuadro.getX(), i *
tamCuadro.getY()));
        }
    }
```

```
    this.setLoop = function (v) {
        loopanim = v;
    };
    this.setVisible = function (v) {
        visible = v;
    };
    this.getPos = function () {
        return pos;
    };
    this.setPos = function (p) {
        pos = p;
    };
    this.getTam = function () {
        return tam;
    };
    this.setTam = function (t) {
        tam = t;
    };
    this.dibujar = function (c) {
        if (new Date().getTime() - ti > animacion.getIntervalo()) {
            ti = new Date().getTime();
            indice++;
            if (indice === animacion.getCuadros().length) {
                if (loopanim) {
                    indice = 0;
                } else {
                    visible = false;
                }
            }
        }
        if (visible) {
            var ca = cuadros[animacion.getCuadros()[indice]];
            c.drawImage(img.getImagen(), ca.getX(), ca.getY(),
                    tamCuadro.getX(), tamCuadro.getY(),
                    pos.getX(), pos.getY(), tam.getX(), tam.getY());
        }
    };
}

function Animacion(cuadros, intervalo) {
    this.getCuadros = function () {
        return cuadros;
    };
    this.getIntervalo = function () {
        return intervalo;
    };
}
```

```
function Textura(ruta) {
    var img = new Image();
    img.src = ruta;
    var tam = null;
    this.getImagen = function () {
        return img;
    };
    this.getTam = function () {
        return tam;
    };
    img.onload = function () {
        tam = new Punto2D(img.width, img.height);
    };
}
function Punto2D(px, py) {
    var x = px, y = py;
    this.getX = function () {
        return x;
    };
    this.setX = function (x) {
        this.x = x;
    };
    this.getY = function () {
        return y;
    };
    this.setY = function (y) {
        this.y = y;
    };

    rnd = function (min, max) {
        return Math.floor((Math.random() * max) + min);
    };
}
```

Código HTML y JavaScript utilizado en la Animación Paletas.

```
<!DOCTYPE html>

<html>
    <head>
        <title>Animación Paletas</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script src="vision.js"></script>
    </head>
    <body>
```

```
        <div id="principal" style="float: left;
             width: 100%;">
            <canvas id="lienzo" onmousedown="mouse(event)"></canvas>
        </div>
    </body>
</html>


var c;
var canvas;
var principal;
var fondo = new Textura("img/sweet-drop-bg.png");
var paleta;
var paletas =
        [new Sprite(new Textura("img/bomb-pop.png"),
                    new Textura("img/bomb-pop-broken.png"),
                    new Punto2D(100, 100)),
            new Sprite(new Textura("img/popsicle.png"),
                    new Textura("img/popsicle-broken.png"),
                    new Punto2D(100, 100)),
            new Sprite(new Textura("img/push-up.png"),
                    new Textura("img/push-up-broken.png"),
                    new Punto2D(100, 100)),
            new Sprite(new Textura("img/creamsicle.png"),
                    new Textura("img/creamsicle-broken.png"),
                    new Punto2D(100, 100))
        ];

function mouse(e) {
    var x = e.clientX;
    var y = e.clientY;
    if (x >= paleta.getPos().getX() && x <= paleta.getPos().getX() +
paleta.getTx1().getTam().getX() &&
            y >= paleta.getPos().getY() && y <= paleta.getPos().getY() +
paleta.getTx1().getTam().getY()) {
        paleta.finalizar();
    }
}
function Sprite(tx1, tx2, pos) {
    var ti;
    var fin = false;
    this.getTx1 = function () {
        return tx1;
    };
    this.getTx2 = function () {
        return tx2;
    };
    this.getPos = function () {
```

```
        return pos;
    };
    this.setPos = function (x, y) {
        pos = new Punto2D(x, y);
    };
    this.dibujar = function (c) {
        if (pos.getY() < canvas.height  - 200) {
            c.drawImage(tx1.getImagen(), pos.getX(), pos.getY());
            if (!fin) {
                pos = new Punto2D(pos.getX(), pos.getY() + 5);
                ti = new Date().getTime();
            } else {
                this.finalizar();
            }
        } else {
            c.drawImage(tx2.getImagen(), pos.getX(), pos.getY());
            this.finalizar();
        }
    };
    this.finalizar = function () {
        fin = true;
        if (new Date().getTime() - ti > 500) {
            this.recrear();
        }
    };
    this.recrear = function () {
        var n = rnd(0, 4);
        var p = paletas[n];
        paleta = new Sprite(p.getTx1(), p.getTx2(),
                new Punto2D(rnd(50, canvas.width -
p.getTx1().getTam().getX() * 2), 0));
    };
}
function Textura(ruta) {
    var img = new Image();
    img.src = ruta;
    var tam = null;
    this.getImagen = function () {
        return img;
    };
    this.getTam = function () {
        return tam;
    };
    img.onload = function () {
        tam = new Punto2D(img.width, img.height);
    };
}
function Punto2D(x, y) {
```

```
    this.getX = function () {
        return x;
    };
    this.setX = function (x) {
        this.x = x;
    };
    this.incX = function (ix) {
        this.x += ix;
    };
    this.getY = function () {
        return y;
    };
    this.setY = function (y) {
        this.y = y;
    };
    this.incY = function (iy) {
        this.y += iy;
    };
}
function rnd(min, max) {
    return Math.floor((Math.random() * max) + min);
}
window.onload = function () {
    principal = document.getElementById("principal");
    canvas = document.getElementById("lienzo");
    c = canvas.getContext("2d");
    paleta = new Sprite(new Textura("img/bomb-pop.png"),
            new Textura("img/bomb-pop-broken.png"),
            new Punto2D(100, 100));
    paleta.recrear();
    setInterval(function () {
        principal.clientheight = window.innerHeight - 10;
        canvas.width = principal.clientWidth;
        canvas.height = principal.clientheight;
        c.drawImage(fondo.getImagen(), 0, 0, canvas.width, canvas.height);
        paleta.dibujar(c);
    }, 1);
};
```

Código HTML Y JavaScript utilizado en la Animación Gotas.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Animación Gotas</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="css/bootstrap.css">
        <script src="js/jquery-1.9.1.js"></script>
        <script src="js/bootstrap.js"></script>
        <script src="js/java.js"></script>
        <script src="js/agua.js"></script>
        <style>
        </style>
    </head>
    <body>
        <div>
            <canvas id="lienzo" width="1600" height="1200"></canvas>
        </div>
    </body>
</html>



$(document).ready(function () {
    new Main().repaint();
});
```

```
class Main {
    constructor() {
        this.canvas = $("#lienzo")[0];
        this.g = this.canvas.getContext('2d');
        this.lienzo = new Lienzo(this.canvas.width, this.canvas.height);
        this.lienzo.padre = this;
    }
    repaint() {
        this.g.drawImage(this.lienzo.canvas, 0, 0);
    }
}
class Lienzo extends Componente {
    constructor(w, h) {
        super(new Point(0, 0), new Dimension(w, h));
        var circulos = [];
        circulos.push(new Circulo(new Point(130, 540), "c1"));
        circulos.push(new Circulo(new Point(80, 320), "c2"));
        circulos.push(new Circulo(new Point(30, 0), "c3"));

        circulos.push(new Circulo(new Point(460, 30), "c1"));
        circulos.push(new Circulo(new Point(410, 140), "c2"));
        circulos.push(new Circulo(new Point(360, 340), "c3"));

        circulos.push(new Circulo(new Point(830, 540), "c1"));
        circulos.push(new Circulo(new Point(780, 320), "c2"));
        circulos.push(new Circulo(new Point(730, 0), "c3"));
```

```
    var instance = this;
    var audio1 = new Audio('sfx/gota.mp3');
    var audio2 = new Audio('sfx/marimba.mp3');
    circulos.forEach(function (o) {
        o.visible = false;
        o.opaco = true;
        instance.add(o);
    });
    $("#lienzo").click(function () {
        instance.opaco = true;
        instance.iniciar(circulos, audio1, audio2);
    });
}

paint(g) {
    this.componentes.forEach(function (c) {
        g.drawImage(c.canvas, c.x, c.y, c.w, c.h);
    });
    if (!this.opaco) {
        g.font = "20px Georgia";
        g.fillText("Click para iniciar", 300, 300);
    }
}

iniciar(circulos, audio1, audio2) {
    var i = 0;
    var j = 0;
```

```
setIntervalX(function () {
    circulos[i].visible = true;
    circulos[i].repaint();
    audio1.play();
    i++;
}, 1000, circulos.length, function () {
    switch (j) {
        case 0:
            circulos[0].visible = false;
            circulos[0].repaint();
            circulos[3].visible = false;
            circulos[3].repaint();
            circulos[6].visible = false;
            circulos[6].repaint();
            break;
        case 1:
            circulos[1].visible = false;
            circulos[1].repaint();
            circulos[4].visible = false;
            circulos[4].repaint();
            circulos[7].visible = false;
            circulos[7].repaint();
            break;
        case 2:
            circulos[2].visible = false;
            circulos[2].repaint();
            circulos[5].visible = false;
```

```
                    circulos[5].repaint();
                    circulos[8].visible = false;
                    circulos[8].repaint();
                    break;
            }
            j++;
            audio2.play();
        });
    }
}
class Circulo extends Componente {
    constructor(pos, tipo) {
        super(pos, getDim(tipo));
        this.textura = new Image();
        var instance = this;
        this.textura.onload = function () {
            instance.repaint();
        };
        this.textura.src = "img/" + tipo + ".png";
    }
    paint(g) {
        try {
            g.drawImage(this.textura, 0, 0, this.w, this.h);
        } catch (e) {

        }
    }
```

```
}


function getDim(tipo) {
    switch (tipo) {
        case "c1":
            return new Dimension(78, 78);
            break;
        case "c2":
            return new Dimension(190, 190);
            break;
        case "c3":
            return new Dimension(301, 301);
            break;
    }
    return new Dimension(0, 0);
}
```

Código HTML y JavaScript utilizado en la Animación Cilindro.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Animación Cilindro</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="css/bootstrap.css">
        <script src="js/jquery-1.9.1.js"></script>
```

```
        <script src="js/bootstrap.js"></script>
        <script src="js/java.js"></script>
        <script src="js/cilindro.js"></script>
        <style>
        </style>
    </head>
    <body>
        <div>
            <canvas id="lienzo" width="1600" height="1200"></canvas>
        </div>
    </body>
</html>



$(document).ready(function () {
    new Main();
});

class Main {
    constructor() {
        this.canvas = $("#lienzo")[0];
        this.g = this.canvas.getContext('2d');
        this.lienzo = new Lienzo(this.canvas.width, this.canvas.height);
        this.lienzo.padre = this;
    }
    repaint() {
        this.g.drawImage(this.lienzo.canvas, 0, 0);
```

```
    }
}

class Lienzo extends Componente {
    constructor(w, h) {
        super(new Point(0, 0), new Dimension(w, h));
        this.opaco = true;
        var ci = new Cilindro();
        this.add(ci);
        var bases = [];
        bases.push(new Base(new Point(0, 150), new Dimension(200, 30)));
        bases.push(new Base(new Point(200, 200), new Dimension(200, 30)));
        bases.push(new Base(new Point(400, 300), new Dimension(200, 30)));
        bases.push(new Base(new Point(600, 400), new Dimension(200, 30)));
        bases.push(new Base(new Point(800, 500), new Dimension(200, 30)));
        var instance = this;
        bases.forEach(function (b) {
            instance.add(b);
        });
        setInterval(function () {
            ci.rotar(1);
            var i = 0;
            bases.forEach(function (b) {
                if (ci.hasColision(b)) {
                    i++;
                }
            });
```

```
            if (i === 0) {
                ci.y += 5;
                if (ci.y > 1000) {
                    ci.y = 0;
                    ci.x = 0;
                }
            } else {
                ci.x += 2;
            }
        }, 1);
    }
}
class Cilindro extends Componente {
    constructor() {
        super(new Point(0, 0), new Dimension(150, 150));
        this.textura = new Image();
        var instance = this;
        this.angulo = 0;
        this.textura.onload = function () {
            instance.repaint();
        };
        this.textura.src = "img/cilindro.png";
    }

    paint(g) {
        try {
            g.translate(this.w / 2, this.h / 2);
```

```
            g.rotate(this.angulo * Math.PI / 180);
            g.translate(-this.w / 2, -this.h / 2);
            g.drawImage(this.textura, 0, 0, this.w, this.h);
        } catch (e) {

        }
    }
    rotar(angulo) {
        this.angulo = angulo;
        this.repaint();
    }
}

class Base extends Componente {
    constructor(location, dimension) {
        super(location, dimension);
    }

    paint(g) {
        g.fillStyle = "black";
        g.fillRect(0, 0, this.w, this.h);
    }
}
```

Código HTML y JavaScript utilizado en la Animación Círculos.

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <title>Animación Circulos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="css/bootstrap.css">
        <script src="js/jquery-1.9.1.js"></script>
        <script src="js/bootstrap.js"></script>
        <script src="js/java.js"></script>
        <script src="js/circulos.js"></script>
        <style>
        </style>
    </head>
    <body>
        <div>
            <canvas id="lienzo" width="1600" height="1200"></canvas>
        </div>
    </body>
</html>
```

```
$(document).ready(function () {
    var lienzo = new Lienzo("lienzo");
});
```

```
class Lienzo {

    constructor(id) {

        this.canvas = $("#" + id)[0];

        this.c = this.canvas.getContext('2d');

        this.circulos = [];

        var chico = 20, mediano = 30, grande = 50;

        this.circulos.push(new Circulo(this.c, 115, 95, grande, "black"));

        this.circulos.push(new Circulo(this.c, 312, 65, chico, "black"));

        this.circulos.push(new Circulo(this.c, 470, 100, grande, "black"));

        this.circulos.push(new Circulo(this.c, 668, 120, grande, "black"));

        this.circulos.push(new Circulo(this.c, 56, 206, chico, "black"));

        this.circulos.push(new Circulo(this.c, 184, 214, mediano,
"black"));

        this.circulos.push(new Circulo(this.c, 290, 176, mediano,
"black"));

        this.circulos.push(new Circulo(this.c, 455, 236, chico, "black"));

        this.circulos.push(new Circulo(this.c, 590, 240, mediano,
"black"));

        this.circulos.push(new Circulo(this.c, 110, 335, grande, "black"));

        this.circulos.push(new Circulo(this.c, 235, 360, chico, "black"));

        this.circulos.push(new Circulo(this.c, 333, 324, mediano,
"black"));

        this.circulos.push(new Circulo(this.c, 482, 354, grande, "black"));

        this.circulos.push(new Circulo(this.c, 685, 348, mediano,
"black"));


        for (var i = 0; i < 20; i++) {

            do {

                var tmp = new Circulo(this.c, rnd(1000), rnd(600), mediano,
"black");
```

```
        var j = 0;
        this.circulos.forEach(function (o) {
            if (o.colision(tmp.x, tmp.y)) {
                j++;
            }
        });
    } while (j > 0);
    this.circulos.push(tmp);
}

this.cursor = new Circulo(this.c, 0, 0, 5, "green");
this.redibujar();
var audio = new Audio('sfx/musica.mp3');
var play = false;
var lienzo = this;
$('#' + id).click(function (e) {
    if (!play) {
        play = true;
        audio.play();
    }
    lienzo.cursor.x = e.pageX;
    lienzo.cursor.y = e.pageY;
    lienzo.redibujar();
    lienzo.circulos.forEach(function (circulo) {
        if (circulo.click(e.pageX, e.pageY)) {
            circulo.cambiarColor("red");
            lienzo.redibujar();
```

```
                }
            });
        });
    }
    redibujar() {
        this.c.fillStyle = "white";
        this.c.fillRect(0, 0, this.canvas.width, this.canvas.height);
        var final = 0;
        this.circulos.forEach(function (circulo) {
            circulo.dibujar();
            if (circulo.color === "red") {
                final++;
            }
        });
//        this.cursor.dibujar();
        if (final === 14) {
            var lienzo = this;
            setInterval(function () {
                getCirculo().r -= 0.1;
                lienzo.redibujar();
            }, 1);
        }
        function getCirculo() {
            lienzo.circulos.forEach(function (circulo) {
                if (circulo.r > 0) {
                    return circulo;
                }
```

```
            });
        }
    }
}

class Circulo {
    constructor(context, x, y, r, color) {
        this.context = context;
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    cambiarColor(color) {
        this.color = color;
    }
    dibujar() {
        this.context.strokeStyle = this.color;
        this.context.fillStyle = this.color;
        this.context.beginPath();
        this.context.arc(this.x, this.y, this.r, 0, 2 * Math.PI, false);
        this.context.fill();
        this.context.stroke();
    }
    click(x, y) {
        var dist = Math.sqrt(Math.pow(this.x - x, 2) + Math.pow(this.y - y,
2));
        return  dist <= this.r;
```

```
    }


    colision(x, y) {
        var dist = Math.sqrt(Math.pow(this.x - x, 2) + Math.pow(this.y - y,
2));
        return  dist <= this.r * 2+30;
    }
}


function rnd(max) {
    return Math.floor(Math.random() * max);
}
```

Código HTML y JavaScript utilizado en la Animación Líneas.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Animación Lineas</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="css/bootstrap.css">
        <script src="js/jquery-1.9.1.js"></script>
        <script src="js/bootstrap.js"></script>
        <script src="js/java.js"></script>
        <script src="js/lineas.js"></script>
        <style>
```

```
        </style>
    </head>
    <body>
        <div>
            <canvas id="lienzo" width="1600" height="1200"></canvas>
        </div>
    </body>
</html>

$(document).ready(function () {
    new Main().repaint();
});

class Main {
    constructor() {
        this.canvas = $("#lienzo")[0];
        this.g = this.canvas.getContext('2d');
        this.lienzo = new Lienzo(this.canvas.width, this.canvas.height);
        this.lienzo.padre = this;
    }
    repaint() {
        this.g.drawImage(this.lienzo.canvas, 0, 0);
    }
}
class Lienzo extends Componente {
    constructor(w, h) {
        super(new Point(0, 0), new Dimension(w, h));
```

```
    this.opaco = true;
    var barras = [];
    for (var i = 0; i < 5; i++) {
        barras.push(new Barra(new Point(i * 200, 0)));
    }
    var instance = this;
    barras.forEach(function (o) {
        o.opaco = true;
        instance.add(o);
    });
    for (var i = 0; i < 5; i++) {
        this.add(new Barra(new Point(i * 200 + 100, 0)));
    }
    var audio = new Audio('sfx/guitarra.mp3');
    $("#lienzo").click(function () {
        instance.iniciar(barras, audio);
    });
    this.repaint();
}
iniciar(barras, audio) {
    var i = 0;
    audio.play();
    barras[i].visible = false;
    barras[i].repaint();
    i++;
    setIntervalY(function () {
        barras[i].visible = false;
```

```
            barras[i].repaint();
            i++;
        }, 1000, barras.length-1);
    }
}

class Barra extends Componente {
    constructor(pos) {
        super(pos, new Dimension(100, 600));
    }
    paint(g) {
        g.fillStyle = "black";
        g.fillRect(0, 0, this.w, this.h);
    }
}
```