ISBN: 978- 99961-50-24-1

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA – FEPADE

DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL

PROGRAMA DE INVESTIGACIÓN APLICADA

INFORME FINAL DE INVESTIGACIÓN

"UTILIZACIÓN DE LA REALIDAD AUMENTADA PARA LA CREACIÓN DE ESCENARIOS EDUCATIVOS"

SEDES Y ESCUELAS PARTICIPANTES: **ESCUELA DE INGENIERÍA ELÉCTRICA**
CENTRO REGIONAL MEGATEC ZACATECOLUCA

AUTOR: **TÉC. JOSÉ ANTONIO HENRÍQUEZ CHAVARRÍA**

ZACATECOLUCA, ENERO 2015

ISBN: 978- 99961-50-24-1

ESCUELA ESPECIALIZADA EN INGENIERÍA ITCA – FEPADE

DIRECCIÓN DE INVESTIGACIÓN Y PROYECCIÓN SOCIAL

PROGRAMA DE INVESTIGACIÓN APLICADA

INFORME FINAL DE INVESTIGACIÓN

"UTILIZACIÓN DE LA REALIDAD AUMENTADA PARA LA CREACIÓN DE ESCENARIOS EDUCATIVOS"

SEDES Y ESCUELAS PARTICIPANTES: **ESCUELA DE INGENIERÍA ELÉCTRICA**
CENTRO REGIONAL MEGATEC ZACATECOLUCA

AUTOR: **TÉC. JOSÉ ANTONIO HENRÍQUEZ CHAVARRÍA**

ZACATECOLUCA, ENERO 2015

Autor
Téc. José Antonio Henríquez Chavarría

FICHA CALCOGRÁFICA

003.3
H519u Henríquez Chavarría, José Antonio.
Utilización de la realidad aumentada para la creación de escenarios
sv educativos / José Antonio Henríquez Chavarría. - 1ª ed. – San Salvador, El Salvador: ITCA Editores, 2015.
44 p. : il. ; 28 cm.

ISBN: 978- 99961-50-24-1

1. Realidad virtual. 2. Simulación por computadores – Enseñanza. I. Título.

Sitio web: www.itca.edu.sv
Correo electrónico: bibliotecologos@itca.edu.sv
PBX: (503) 2132 – 7400 / FAX: (503) 2132 – 7423

Tiraje: 16 ejemplares
ISBN: 978- 99961-50-24-1
Año 2015

TABLA DE CONTENIDO

CONTENIDO **Pág.**

1. INTRODUCCIÓN

La tecnología como elemento o medio para lograr mejores aprendizajes, es una de las ventajas de las nuevas herramientas en el área de desarrollo de software y hardware.

La tecnología de Realidad Aumentada se encuentra en una etapa de crecientes descubrimientos y aplicaciones, las cuales aplicadas de forma adecuada y precisa a diversas situaciones en el aula, permitirán mejorar el nivel de aprendizaje de los estudiantes; además de brindar mejoras o productos de tecnología propios. Tal es el caso del presente proyecto, el cual busca documentar herramientas de la realidad aumentada, que se puedan aplicar en el desarrollo de futuros proyectos, que contemplen la creación de escenarios educativos en realidad aumentada.

2. PLANTEAMIENTO DEL PROBLEMA

6.1 DEFINICIÓN DEL PROBLEMA

La enseñanza técnica plantea requerimientos de equipo y acceso a instalaciones que con frecuencia no se posee, las instituciones educativas en general necesitan utilizar los recursos disponibles para desempeñar sus labores educativas de una forma efectiva.

Es sabido que la enseñanza técnica es más efectiva cuando se lleva al estudiante a un entorno práctico similar o igual al entorno que viviría en una empresa, por lo tanto las instituciones técnicas deben tener laboratorios que tengan condiciones, equipos y ambientes similares al que encontrará en una empresa.

Lo anterior trae una situación de constante búsqueda creativa e innovadora de formas de disponer equipo y optimizar el uso de las tecnologías existentes para brindar mejores servicios educativos

Por ejemplo en el MEGATEC Zacatecoluca, la carrera de Técnico en Logística global, no posee instalaciones y equipo para analizar el tema de distribución de transporte; el Técnico en Mantenimiento de computadoras, no tiene equipos de redes suficientes para comprender configuraciones de redes inalámbricas; en Técnico en Electrónica requiere espacios para comprender instalaciones de equipo y redes eléctricas. Lo anterior plante a la institución la necesidad de entornos donde se puedan “simular” prácticas y/o en los cuales se requiere el uso de equipos y herramientas que no se cuenta en la institución.

Al no poseer los centros de educación todos los escenarios probables, en el que se desempeñará el futuro profesional, se hace necesario buscar los medios para su obtención; la realidad aumentada, promete ser una técnica que puede apoyar en disminuir la brecha del mundo real a lo aplicado en las aulas, al menos en ambientes emulados.

2.2 ESTADO DE LA TÉCNICA.

Hoy en día las aplicaciones de realidad aumentada se basan en 2 tipos:

A. Reconocimiento de marcas o patrones.

Comúnmente denominadas **patterns** los cuales son símbolos especiales los cuales se imprimen en un papel y se ponen enfrente de la cámara el reconocimiento de estos patrones[1] provoca que se realicen las acciones indicadas en pantalla.

[1] http://www.realidad-aumentada.eu/tecnicas-de-la-realidad-aumentada/

Fig. 1 Patrón de reconocimiento

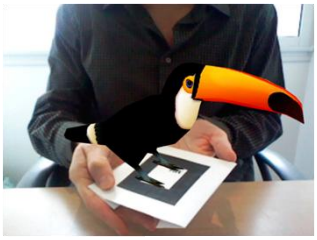


Fig. 2 Reconocimiento del patrón y muestra con la acción asociada que dispara una

Esta es la técnica más común por su rápida implementación ya que se puede encontrar diferentes proyectos de código cerrado los cuales nos dan las herramientas básicas, patrones más comunes y diseños con los cuales se pueden trabajar pero tienen una limitante en cuanto a los escenarios que se pueden desarrollar, porque se limitan en la mayoría de casos solo a representar objetos estáticos en 3D en pantalla sobre los objetos reales. Diferentes aplicaciones utilizan esta técnica entre ellas podemos mencionar libros con contenidos estáticos.

Fig. 3. Aplicación de realidad aumentada, en el proceso de enseñanza

B. Reconocimiento de imágenes y Comparación.

Entre la mejora a la realidad aumentada se utiliza el reconocimiento de imágenes para comparar puntos de interés, los cuales permiten ampliar los elementos con los cuales se pueden realizar las interacciones y/o disparar las acciones en nuestro escenario virtual y ya no estamos limitados al uso de patrones en blanco y negro. Básicamente para aplicar esta técnica primero tenemos que definir el elemento (un objeto común, una tarjeta de presentación un logo) el cual será reconocido por medio de la técnica de verificación de los puntos de interés y al momento que la cámara detecte dicho objeto se realizan las acciones que se han programado.

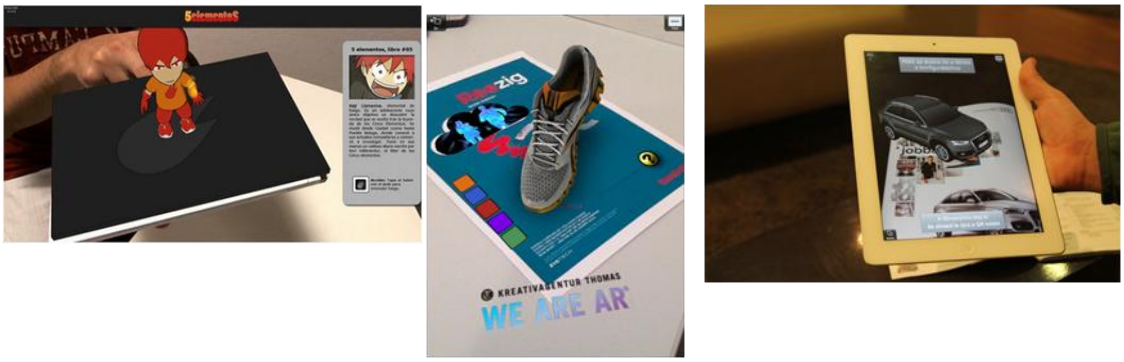


Fig. 4. Aplicación de realidad aumentada, en el área empresarial

Esta técnica nos permite mejorar la interacción con otros objetos y amplía las posibilidades, pero debemos mencionar que la mayor desventaja es por los recursos que se necesitan para poder desarrollar las interacciones.

Estas son las dos técnicas actualmente de mayor uso, muchas universidades y comunidades de investigación están mejorando los algoritmos de reconocimiento de imágenes y nuevas librerías que permitan una mejor integración con los entornos informáticos actuales.

Fig. 5 Simulador usado por BMW para entrenamiento del personal técnico de talleres

Actualmente la realidad aumentada se utiliza en muchas áreas desde el entretenimiento hasta procesos industriales[2], la educación y medicina son las dos aéreas por las cuales se está apostando a la implementación y aprovechamiento de dicho recurso ya que se pueden simular procesos de riesgo sin poner en peligro a los pacientes, conocer instrumentaciones nuevas o manipulación de nuevos equipos que ayuden en las cirugías y la rehabilitación de pacientes.

[2]http://www.popsci.com/scitech/article/2009-09/bmw-developing-augmented-reality-help-mechanics

Fig.6 Simulación de Cirugía con Realidad Aumentada

Fig. 7 Tratamiento de pacientes en Rehabilitación

En educación permite realizar escenarios controlados que permitan realizar prácticas en las cuales se necesiten equipos y/o herramientas con las cuales no cuenta la institución y son difíciles de adquirir, simular prácticas de taller en la cuales permitan mejorar o implementar nuevas técnicas para realizar los procesos.

Muchas compañías están invirtiendo en desarrollos e investigación acerca de aplicaciones de Realidad Amentada entre ellas Google.

Fig. 8 Dispositivo, de proyecto de Google Glass.

Esta se encuentra desde hace un año en la fase beta de su proyecto Google Glass[3], el cual es un dispositivo similar a un par de anteojos, los cuales integran una pequeña pantalla la cual servirá para interactuar o mostrar información de lugares turísticos, información del clima, ubicación de lugares, búsquedas en internet para ampliar la información de una ubicación o dirección.

[3]http://www.google.com/glass/start/what-it-does/

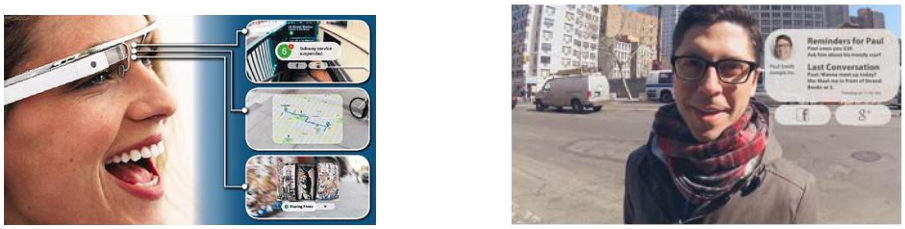


Fig. 9 Aplicación de Google Glass, para información turistica

Este aporte que está brindando esta compañía provee de una mejor técnica, nuevos elementos de hardware y la mejora el software para poder generar más y mejores aplicaciones que permitan una mejor aplicación y masificación de lo que es la implementación de realidad aumentada.

El sector de los videojuegos no se queda afuera del uso y aplicaciones de realidad aumentada y el mejor exponente en este caso es Microsoft con su tecnología Kinect.[4] Kinect permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, objetos e imágenes. El dispositivo tiene como objetivo primordial aumentar el uso de la Xbox 360, más allá de la base de jugadores que posee en la actualidad.

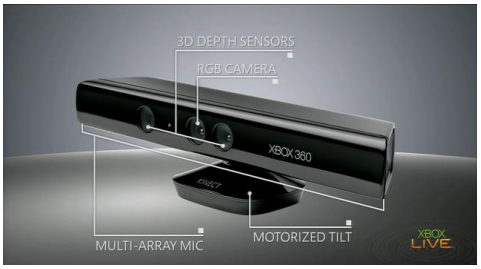


Fig.10 Kinect

Esto permite una experiencia entre un video jugador para poder controlar con todo su cuerpo las acciones que se deben de desarrollar en el juego, pero las comunidades de desarrollo han puesto manos a la obra y han desarrollado nuevas aplicaciones que no solo sean para juegos sino que permitan llevar la realidad aumentada a un punto de mejor aplicación e interacción con el usuario, unos ejemplos de los desarrollos son:

- Una simulación de un camuflaje para la persona lo cual lo vuelve invisible para la cámara[5].
- Un mapa en tres dimensiones para poder demostrar el uso en la construcción, recursos hídricos y otras capas para poder interactuar con el terreno.

[4]http://www.xbox.com/es-ES/Kinect
[5]http://www.fayerwayer.com/2010/12/con-kinect-puedes-jugar-a-ser-un-%E2%80%9Cdepredador%E2%80%9D/

- Probador virtual en tiendas de ropa[6].

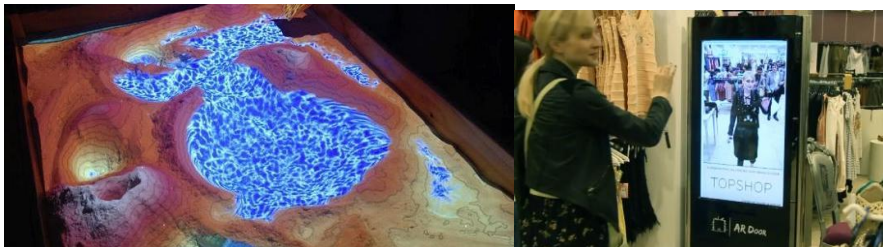


Fig. 11 Aplicaciones de realidad aumentada en mercadeo

Otro avance del uso y mejoras de realidad aumentada es el uso de los Smartphone[7] ya que hoy muchas compañías hacen uso de aplicaciones, con la cuales se puede ver información ampliada de los productos a comprar, activar promociones y ver publicidad extra de un producto, tiendas cercanas que proveen descuentos, cupones virtuales y otras sencillas aplicaciones que amplían la experiencia entre las personas y su entorno virtual.

Fig. 12 Aplicaciones de realidad aumentada en los Smartphone

Hay muchos proyectos de software que permiten el uso de realidad aumentada ente los cuales podemos encontrar muchos en la Web estos pueden ser de pago o gratuitos, muchos implementan todas las características que se mencionan, pero para mejorar la experiencia en el desarrollo debemos no solo tener un software que permita crear escenarios de realidad aumentada, sino que debemos de complementar con los dispositivos de hardware los cuales permitan ampliar la experiencia y aplicación de realidad aumentada.

[6]http://www.puntogeek.com/2011/05/29/probador-virtual-con-kinect/
[7]http://www.realidad-aumentada.eu/tecnicas-de-la-realidad-aumentada/

2.3 JUSTIFICACIÓN

Los avances tecnológicos permiten aprovechar ventajas de los equipos y tecnología disponible a las organizaciones; en otras palabras requieren el uso eficiente y efectivo de lo que se posee.

En la Escuela Especializada en Ingeniería, se cuenta con un equipo base para trabajar con los estudiantes de cada carrera técnica, sin embargo este equipo e instalaciones no son suficientes para satisfacer las necesidades y capacidades demostrativas, prácticas y de entrenamiento que se necesita para llevar la enseñanza al más alto nivel de aprendizaje.

Dado que la *Realidad Aumentada* es una técnica o herramienta moderna para uso y aplicación de tecnologías existentes en nuestro medio, se considera necesario y oportuno analizar sus capacidades, diseñar modelos y proponer ambientes educativos que aprovechen el uso de esta tecnología para la enseñanza técnica.

Siendo la técnica de realidad aumentada, una herramienta no desarrollada en el ámbito de educación superior, por falta de un dominio en su programación; se hace necesario, disponer de una base de herramientas que se deben de manejar para el desarrollo de escenarios educativos; ya que será la base de partida, para futuros programadores, que desearán aplicar la realidad aumentada en ambiente de aprendizajes.

3 OBJETIVOS.

3.1 OBJETIVO GENERAL

Aplicar la técnica de realidad aumentada, para el uso de la enseñanza técnica.

3.2 OBJETIVOS ESPECÍFICOS:

- Conocer y experimentar diferentes entornos de realidad aumentada que puedan ser utilizados para emular e inter-actuar en situaciones de aprendizaje del área técnica.
- Seleccionar herramientas de software aplicables a realidad aumentada, para ser utilizadas en la construcción de situaciones de aprendizaje.
- Consolidar un repositorio de códigos de herramientas de software, para el desarrollo de situaciones de aprendizaje de realidad aumentada.

4 MARCO TEÓRICO

6.2 ACTORES Y PROVEEDORES PRINCIPALES EN EL USO DE TECNOLOGÍA DE REALIDAD AUMENTADA.

La RA (Realidad Aumentada), es una tecnología que tiene actualmente todo un mercado en desarrollo donde interactúan diversos entes involucrados, entre ellos están:

Los fabricantes de dispositivos:

Son los fabricantes del hardware de los dispositivos sobre los que es posible acceder a este tipo de servicios. Se trata de los fabricantes de smartphones, de PC, tablets y consolas y, en el caso de la realidad aumentada, de los fabricantes de gafas especiales o cascos que incorporan pantallas, proyectores y en un futuro de los lentes que permitirán ver directamente la información digital superpuesta.

Los desarrolladores del software de realidad aumentada

Los agentes que desarrollan las aplicaciones que activan los datos que complementan la visión del entorno físico y gracias a los que es posible disponer el mundo virtualmente aumentado y así enriquecer los objetos de la vida real e interactuar con ellos de una manera radicalmente diferente.

Mundo digital y digitalizado:

Las empresas proveedoras de contenidos y los usuarios como productores. Son un elemento especialmente relevante son en sí los datos, es decir, el mundo digital y el digitalizado. En los últimos años se ha capturado mucha información sobre el mundo real, por ejemplo datos sobre la localización, descripciones, imágenes, etc. tanto de lugares como de cosas y personas y toda esa información se ha ido almacenando en ***la nube***, lo que ha ido configurando un mundo digital *paralelo*. Ejemplos concretos de esto podrían ser, la W**ikipedia**, que recoge cantidades de información de diferentes temas constituyendo así la mayor enciclopedia del mundo, google earth, que permite ver el mundo a través de imágenes satelitales, mapas, relieves o edificios 3D o las redes sociales Facebook o LinkedIn (esta profesional), que albergan gran cantidad de información de sus usuarios.

La red: operadoras de telecomunicación:

El otro agente esencial para que estas aplicaciones puedan llegar al público es la red, ya que la capacidad de la realidad aumentada radica en poder acceder a la información digital complementaria a la del mundo físico actualizada en tiempo real. El acceso a esta información debe poder realizar se en muchos casos como en las aplicaciones geo-referenciadas desde cualquier lugar y en cualquier momento. Por ello, las operadoras de telecomunicación desempeñan un papel especialmente relevante en este ecosistema.

6.3 APLICACIÓN DE LA REALIDAD AUMENTADA.

Hay diversas formas y aplicaciones de la RA, a continuación se presenta un resumen de sus principales áreas de aplicación actuales:

Realidad aumentada en juegos:

En el año 2000 algunas universidades comenzaron a ver el potencial que podía tener el uso de la realidad aumentada y para su investigación comenzaron a crear réplicas de juegos para el ordenador o las videoconsolas usando esta tecnología. Un juego clásico muy conocido y replicado de este modo es PacMan, que fue implementado por la National University of Singapore, de manera que el jugador podía ser, bien un fantasma o el propio Pac-Man y el laberinto eran las propias calles de Singapur. Para poder jugar, el usuario tenía que disponer de un ordenador portátil, GPS, Bluetooth, wifi, infrarrojos y sensores. Otro juego destacado que ha sido llevado a realidad aumentada, en este caso por Wearable Computer Lab de la University of South Australia, ha sido Quake

Realidad aumentada en enseñanza

El campo de la enseñanza es otro en el que las aplicaciones de realidad aumentada adquieren mucho sentido. En la actualidad, están apareciendo aplicaciones sociales, lúdicas y basadas en la ubicación que muestran un potencial importante para las aplicaciones en este ámbito, tanto para proporcionar experiencias de aprendizaje contextual como de exploración y descubrimiento de la información conectada en el mundo real. Uno de los campos de aplicación de la realidad aumentada son los libros.

Un ejemplo de esta aplicación es el de la alemana Metaio que desarrolla libros que incluyen elementos de este tipo utilizando realidad aumentada basada en el uso de códigos. Los libros se imprimen de manera normal; después de la compra, los consumidores instalan un programa especial en sus computadoras y apuntan al libro con una cámara web para ver las visualizaciones. La tecnología permite que cualquier libro pueda desarrollarse en una edición de realidad aumentada después de publicarlo. En la actualidad, están desarrollando un atlas que contiene visiones 3D de lugares geográficos.

Realidad aumentada en marketing y venta

El marketing y los procesos de venta son las áreas donde más se está aplicando la realidad aumentada. En relación al marketing, área en la que captar la atención es un elemento fundamental, las empresas ven la realidad aumentada como una forma de diferenciarse con respecto a la competencia, ofreciendo al usuario la posibilidad de acceder a experiencias visuales llamativas.

Realidad aumentada en viajes y guías turísticas

Un ejemplo de aplicación es Wikitude15 que permite, con su versión «Travel Guide» gracias a una aplicación instalada en un smartphone, permite detectar qué es lo que se está viendo en cada momento y mostrar la información más relevante sobre el lugar (información histórica, monumentos emblemáticos cercanos, puntos de interés, etc.). Wikitude, utiliza una combinación entre la cámara, la brújula, la conexión a internet y el GPS del teléfono móvil para activar la AR. Con ello, se identifica la posición del usuario y la orientación, después se reciben los datos pertenecientes al objeto enfocado y se muestra en la pantalla sobre la imagen capturada por la cámara. El contenido se extrae de Wikipedia, Qype y Wikitude, y los usuarios pueden añadir información propia.

Realidad aumentada aplicada a procesos de búsquedas

La navegación y las búsquedas por internet, por ejemplo aplicaciones que ayudan a encontrar la parada de autobús más cercana o los cajeros automáticos de la zona, las consultas de médicos, así como las cafeterías y restaurantes, etc.

Realidad aumentada social

Otra aplicación de la realidad aumentada tiene que ver con un uso social. Se trata de mezclar las redes sociales y las interfaces de RA de manera que se satisfaga la necesidad humana de encontrar gente y compartir experiencias e información con familiares, amigos y compañeros.

Realidad aumentada en medicina

Por ejemplo, para un cirujano, puede ser muy importante disponer de tres dimensiones de los órganos y huesos, alrededor de la zona en la que está llevando a cabo una intervención, o también información complementaria como datos del paciente o sobre la operación.

5 METODOLOGÍA DE LA INVESTIGACIÓN

El proceso metodológico para realizar el proyecto, consistió en diversas etapas aplicando en todo momento el método científico.

Etapa 1. Investigación inicial

La investigación documental da los insumos de donde se necesita indagar a profundidad el estado de la técnica, los antecedentes de la investigación y los avances de Realidad aumentada y aplicaciones a la educación. Esto implicó aplicar técnicas como: revisión documental en internet y en libros. Al final de esta etapa se obtuvo información teórica para desarrollar el documento guía y pruebas de herramientas bases, para el desarrollo de la realidad aumentada.

Etapa 2. Investigación de campo

Para el desarrollo del proyecto se desarrollaron fases del Método de Investigación Operativa (INOP). La INOP, es un método de investigación que tiene las siguientes fases:

1. Identificación del problema y análisis de alternativas: se identificaron las probabilidades de aplicación de la realidad aumentada a las necesidades de las aulas de clases en módulos que requieren equipo o ambientes que no están disponibles; teniendo necesidad en todas las carreras de MEGATEC ZACATECOLUCA.

2. Modelado: Se analizaron los componentes de aplicación y los requerimientos de la realidad aumentada y su aplicación en un aula que modele ambientes de aprendizaje reales-virtuales; determinando como una gama de aplicaciones de la realidad aumenta, en los salones de clases.
3. Resolución: Se plantea y diseña una guía que permite implementar o aplicar la realidad aumentada para crear escenarios de realidad aumentada en un aula de clases. El objetivo de dicha guía es servir de base para las necesidades de modelado con realidad aumentada.
4. Presentación de guía*:* Construcción de la guía propiamente dicha, para hacer un simposio de herramientas básicas, para el desarrollo de la realidad aumentada, esto contiene, los códigos base para el poder manejar el Sensor Kinect y controlas sus diferentes métodos de captura*.* Para el desarrollo de la guía se hace uso de las siguientes técnicas y herramientas:

Técnicas de recolección de información a utilizar:

- Revisión documental.
- Entrevista: con docentes para identificar módulos o contenidos que necesiten u permitan aplicación de realidad aumentada; para justificar su aplicación.

Técnicas de programación orientada a objetos:

Durante el desarrollo del proyecto se ha utilizado la Técnica de programación Orientada a Objetos, que es una de las que actualmente mejoran los resultados en el desarrollo y brindan mayor eficiencia al software. Actualmente los software par manejo de imágenes y diversos objetos de programación visual, se implementan utilizando la Programación orientada a objetos ya que facilita la comprensión y manejo de la información de cada objeto del sistema y la programación de las acciones que sobre él se realizan. La programación Orientada a Objetos implementa mecanismos Herencia, polimorfismo, encapsulamiento, para un mejor rendimiento y aprovechamiento de los recursos de los programas creados con esta técnica.

Herramientas y metodologías implementadas en la construcción.

En el desarrollo del aplicativo para la construcción de escenarios de educación basados en realidad aumentada, se necesitan componentes tanto de hardware y software los cuales deben de poseer una compatibilidad perfecta, la cual es parte de una buena implementación, en el caso de estudio se consideraron los siguientes.

Hardware:

1. Microsoft Kinectfor Windows. Se ha optado por este componente de hardware por ser un conjunto de sensores de audio y vídeo permiten la captura de los recursos tanto visuales como de audio para alimentar el aplicativo que será la materia prima para el trabajo del aplicativo.

   En la parte técnica las características del Kinect son las siguientes (estos son datos específicos para la versión para Windows).

   i. Lentes de color y sensación de profundidad
   ii. Micrófono Multi-arreglo.
   iii. Ajuste de inclinación
   iv. campo de visión horizontal 57°
   v. campo de visión vertical 43 °
   vi. rango de inclinación física ±27 °
   vii. rango de profundidad modo normal

0-0,8 metros	Fuera de rango
0.8 – 4 metros	Parámetros normales
4 – 8 metros	Se capturan datos pero no son óptimos

2. Rango de profundidad modo cercano.

0-0,4 metros	Fuera de rango
0.4 – 3 metros	Parámetros normales (mejor calidad de captura a 2 metros)
3 – 8 metros	Se capturan datos pero no son óptimos
> 8 metros	Fuera de rango.

Software:

Para el desarrollo de la plataforma se utiliza:

- Microsoft visual estudio 2010
- Microsoft Kinect SDK 1.7
- Blender (software de modelado 3D)
- Microsoft XNA 4.0

Estos componentes permitirán con investigación y pruebas el poder desarrollar los componentes necesarios para poder integrar el software final que se planea implementar.

6 RESULTADOS.

Por diversas razones de recursos y tiempo para profundizar en el desarrollo del proyecto, no se logró implementar en el aula herramientas de realidad aumentada. Por lo que de forma alternativa, se presenta la base teórica y técnica desarrollada para que sirva de punto de partida en la creación de escenarios de aprendizaje.

USO DE HERRAMIENTAS PARA CREAR APLICACIONES CON KINECT, EN REALIDAD AUMENTADA.

Para el uso de la herramienta de Kinect, primero se deben conocer las bases técnicas de los componentes de hardware; ya que Microsoft, proporciona las librerías de desarrollo para que podamos crear aplicaciones usando el Kinect.

Para poder desarrollar tenemos que tener bases de un lenguaje de programación con el cual podremos codificar las aplicaciones con Kinect, que se adapten a la necesidad en la cual se implementara; para ello podemos desarrollar la aplicaciones con código no manejado para esta opción, optando por el lenguaje nativo, el cual es C++, así también podemos optar por crear aplicaciones manejadas, las cuales se desarrollan con C+ o Visual Basic, además con la salida de la versión 1.8 del SDK, podemos desarrollar interacciones en aplicaciones Web con HTML5 usando JavaScript.

Para crear aplicaciones con el Kinect debemos saber controlar ciertas características las cuales nos ayudaran a hacer más interactivo nuestro desarrollo. Las características son las siguientes:

- Cámara de Color.
- Sensor IR para la Profundidad.
- El Sensor de Esqueleto.
- Los Micrófonos.
- Motor para poder manipular el ángulo.

Para comenzar el desarrollo de aplicaciones con Kinect, el primer paso es detectar e inicializar el Kinect, al poder realizar esta acción, ya tendremos acceso a utilizar cualquiera de las características que mencionamos con anterioridad.

6.4 INICIALIZACIÓN DE KINECT.

Para hacer nuestras aplicaciones y ejemplos podremos utilizar Microsoft Visual Estudio como herramienta de trabajo y esto es porque nos brinda muchas características que nos facilitaran el trabajo de crear aplicaciones con el Kinect.

Para el desarrollo lo primero que debemos tomar en cuenta es contar con el sensor Kinect junto con el adaptador múltiple que nos sirve para poder conectarlo al PC y a la alimentación de corriente, ya que el conector USB del PC no provee el voltaje necesario ni el amperaje que el Sensor Kinect necesita para su correcto funcionamiento, así que es sumamente necesario contar con dicho adaptador, el Kinect For Windows ya provee este accesorio, esto cabe mencionarlo ya que si cuenta con una versión del Sensor Kinect for Xbox 360 dicho accesorio se debe adquirir por separado.

Fig.13 Adaptador de corriente y conector del Kinect

6.5 Creando Un "Hola Mundo Kinect"

Necesitamos hacer:

1. Crear un nuevo proyecto en Visual estudio.
2. Las configuraciones básicas de dicho proyecto serán.
    a. Lenguaje de desarrollo C++.
    b. Tipo de proyecto WPF.
    c. Definimos un nombre para nuestro caso será "***KinectBasico***".
    d. Opcionalmente cambiamos la ubicación de donde se almacena el proyecto.
3. Al cargar nuestro proyecto debemos agregar la referencia de las librerías que utilizaremos para poder hacer uso del Sensor Kinect. en este caso haremos uso del ensamblado de librerías del Kinect SDK 1.8, para agregarlas debemos hacer lo siguiente.
    a. Vamos al explorador de soluciones y damos clic derecho sobre el proyecto, en el menú emergente seleccionamos ***agregar*** y buscamos la opción ***referencia***.
    b. En el cuadro de dialogo vamos a la opción de examinar y buscamos la librería en la carpeta de instalación del SDK generalmente la ruta es la siguiente. ***C:\Program Files\Microsoft SDKs\Kinect\v1.8\Assemblies***. seleccionamos el archivo de la librería y damos clic en ok, con este proceso ya se agregan las librerías necesarias para empezar a programar con el Kinect.
4. Creamos el código para poder inicializar el Kinect.
    a. En el archivo cargado de ***MainWindow.xaml***, seleccionamos el componente ventana y en el explorador de propiedades, nos dirigimos a la pestaña u opción de eventos y buscamos el evento ***loaded***, damos doble clic para agregar un nuevo controlador de eventos ***loaded***. En el archivo ***MainWindow.xaml.cs*** se agrega automáticamente el siguiente código.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
}
```

b. Definimos la referencia de la librería agregándola en la clase en el apartado de la declaración de los ***using*** de dicha clase.

```
using Microsoft.Kinect;
```

c. En la definición del evento que se crea se agregan las siguientes líneas de código.

```
KinectSensor mSensor;
if (KinectSensor.KinectSensors.Count == 0)
{
    MessageBox.Show("No hay Dispositivo Conectado");
    Application.Current.Shutdown();
}

mSensor = KinectSensor.KinectSensors[0];
mSensor.Start();
```

Con esto realizamos una verificación si el sensor está conectado en el sistema, se retorna un mensaje si este no es detectado, si está conectado creamos una variable del tipo ***KinectSensor*** denominada ***mSensor,*** la cual almacena la referencia del objeto de la API almacena en un arreglo el número de Kinects conectados al pc, al tener el objeto utilizamos la función ***Start***() que se encarga de activar el sensor, este pasa a un estado activo con lo cual ya puede recibir datos y esperar activar los sensores para las acciones de cámara, sensor IR micrófonos , etc.

En el sensor podremos ver que se enciende el emisor de IR[8], y con esto verificamos que esta encendido e iniciado el sensor. Este sería nuestro ejemplo más básico para el control del sensor.

[8] Sensor de Infrarrojos

Fig. 14 Kinect Activo y vista de haz de rayos IR del sensor

6.6 INTERACTUANDO CON EL SENSOR: USO DE CÁMARA RGB.

Ahora que podemos iniciar nuestro sensor podemos iniciar sus diferentes componentes para la captura de información, en este caso realizaremos un ejemplo con el cual habilitaremos el sensor de color del Kinect, el cual permite capturar imágenes desde la cámara RGB que posee nuestro dispositivo, esto es útil ya que podremos crear una interfaz de video en tiempo real la cual combinándola con otros elementos del sensor será la base para poder crear Realidad aumentada ya que sobre los elementos capturados de nuestra cámara podremos agregar gráficos, otros videos, animaciones, elementos 3D, etc.

Antes de entrar en la codificación de nuestro ejemplo vamos a analizar unos conceptos que nos ayudaran a comprender como se comporta la captura de los datos por medio del sensor de color del Kinect.

Las imágenes en nuestras pantallas (Pc, teléfono, televisión) están formada por pixeles que son puntos que almacenan y mezclan espacios para los colores los cuales permiten construir las imágenes que nuestro cerebro procesa y entiende.

Por eso debemos tener en consideración que la manipulación de estos elementos es fundamental para el trabajo de Realidad aumentada ya que combinaremos los elementos capturados de la realidad con los elementos que podemos construir digital o virtualmente.

Una característica que cabe resaltar de la combinación de los pixel es que de acuerdo a filtros o forma de ordénalos, determinan un formato de imagen que se puedan procesar, Kinect puede

procesar los siguientes formatos:

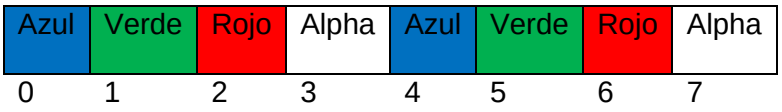
- RGB (red, green, blue).
- YUV (Luminosidad, blue, red).
- Bayer[9].

Estos formatos permiten adecuarse dependiendo la necesidad del desarrollador, el formato más utilizado es el RGB, Kinect en su documentación hace referencia a 2 posibles usos del formato el cual va depender la resolución pantalla y los fotogramas por segundo (fps) que se utilicen o debamos aplicar en nuestra programa.

Entonces cuando usemos como base de captura RGB Kinect devolverá un stream de datos de 32 bits el cual se puede tratar de acuerdo a 2 formatos los cuales son:

- 640 x 480 a 30 fps.
- 1024 x 960 a 12 fps.

En la captura la composición del stream viene dado por el siguiente esquema.



Esquema de composición del stream

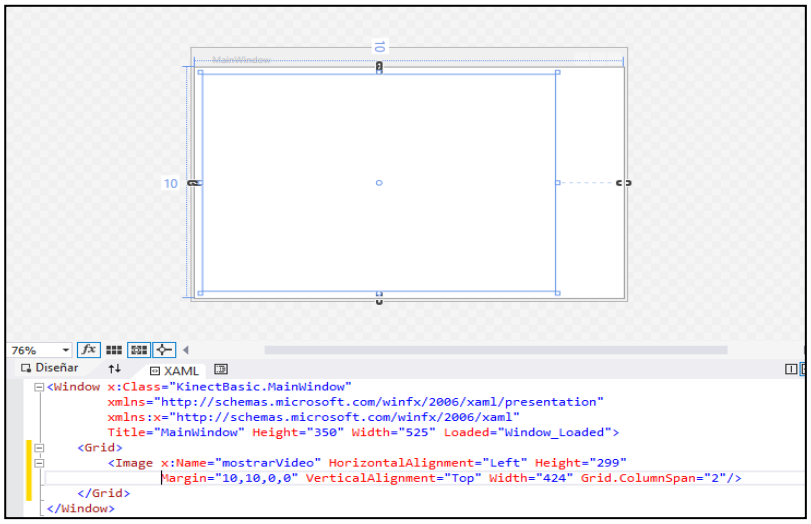
Este esquema nos ayuda a comprender la forma en la cual se capturan en este caso los pixeles en formato RGB, adicionalmente Kinect agrega un espacio del stream para el canal alpha, esto es para manejar un porcentaje de trasparencias.

Cuando el Kinect capture los datos este stream debe ser almacenado en una matriz para poder construir las imágenes que mostraremos en pantalla.

6.7 Capturando Y Mostrando un Stream de Color.

Para realizar la captura del stream de color, creamos un nuevo proyecto el cual aplicaremos los pasos que se hicieron en el ejemplo 1, pero en este caso agregaremos un control ***Image*** de WPF que será nuestro contenedor de la captura y muestra del stream de color del Kinect.

[9] Filtro de color en el cual se sitúan los colores básicos sobre un sensor digital.



Vista de la ventana de diseño

En la pestaña de XAML verificamos que el código se muestre como se muestra a continuación:

```
<Window x:Class="KinectBasic.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525" Loaded="Window_Loaded">
    <Grid>
        <Image x:Name="mostrarVideo" HorizontalAlignment="Left" Height="299"
               Margin="10,10,0,0" VerticalAlignment="Top" Width="424"
Grid.ColumnSpan="2"/>
    </Grid>
</Window>
```

El ***x:Name*** es la propiedad que nos servirá para enlazar el stream que capturemos y se muestre en un formulario de la aplicación y así poder ver en pantalla la captura del stream y así poder darle un tratamiento de acuerdo a lo que necesitemos crear.

Entonces agregamos las siguientes líneas de código teniendo en cuenta que usaremos la base del código del ejemplo 1.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    if (KinectSensor.KinectSensors.Count == 0)
    {
        MessageBox.Show("No hay Dispositivo Conectado");
        Application.Current.Shutdown();
    }
    try
    {
        mSensor = KinectSensor.KinectSensors[0];
        mSensor.Start();
        mSensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
        mSensor.ColorFrameReady += mSensor_ColorFrameReady;
        }
    catch
    {
        MessageBox.Show("No hay Dispositivo Conectado");
        Application.Current.Shutdown();
    }
}
```

En este caso el objeto ***KinectSensor***, que se identifica siempre como ***mSensor***, se tendrá que realizar las siguientes acciones:

1- Iniciar el objeto con la función ***Start***()

2- Habilitar el Objeto Color Stream que nos servirá para almacenar el stream que capture la camara RGB del kinect, esto se hace por medio de la función: ***ColorSteam.Enable()***, adicionalmente se para un parámetro para indicar el formato especifico que deseamos en la captura, para ello debemos utilizar la especificación que nos brinda la API por medio de la sentencia:
ColorImageFormat.RgbResolution640x480Fps30.

3- Se adiciona un controlador del evento para para la captura del flujo de stream, esto se realiza por medio de la instrucción:
ColorFrameReady += mSensor_ColorFrameReady.

Estas acciones permitían que el Kinect esté listo para poder recibir las capturas desde la cámara de color y poder tratarlas en el stream de datos.

Pero debemos de especificar como vamos a procesar el stream y en donde veremos el resultado, en este caso en el controlador de eventos codificaremos directamente las instrucciones para poder ver un resultado directo. Entonces si utilizamos la ayuda de Visual Estudio ya tendremos que tener creada una estructura del controlador del evento: ***ColorFrameReady,*** si no es el caso agregamos el siguiente código.

```
void mSensor_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
       using (ColorImageFrame frameImagen = e.OpenColorImageFrame())
       {
              if (frameImagen == null)
              {
                     return;
              }

              byte[] datosColor = new byte[frameImagen.PixelDataLength];
              frameImagen.CopyPixelDataTo(datosColor);
              mostrarVideo.Source = BitmapSource.Create(
                     frameImagen.Width,frameImagen.Height,
                     96,96,PixelFormats.Bgr32,null,datosColor,
                     frameImagen.Width * frameImagen.BytesPerPixel
                     );
       }
}
```

Con este código en el controlador lo que hacemos es usar una definición de directiva ***using***, que nos ayuda para poder controlar los objetos que deben ser liberados de forma obligatoria cuando finaliza su uso, en este caso el objeto ***ColorImageFrame***.

1- Se comprueba que el objeto no este nulo.
2- Creamos un arreglo de byte para poder almacenar los pixeles que la cámara este capturando.
3- Con la función ***CopyPixelDataTo*** se copia cada pixel para almacenarlo en el arreglo de bytes que creamos antes.
4- Al objeto en nuestro formulario Image debemos de pasar las instrucciones necesarias para poder mostrar los datos que tenemos en el arreglo de bytes para ello en su propiedad ***Source***, debemos indicar a que se igualara esa fuente de información en este caso será un objeto del tipo ***BitmapSource***, el cual debemos pasar los parámetros necesarios para poder crear la imagen, que mostraremos en dicho control los parámetros que pasamos son los siguientes.
    - Ancho de la bitmap a mostrar o crear.
    - Alto de la bitmap a mostrar o crear.
    - Valor del dpi horizontal (DPI)[10]
    - Valor del dpi en la vertical(DPI)
    - El formato de los pixeles
    - Paleta de color del bitmap
    - Los bytes que crearan el bitmap

[10] Puntos por pulgadas

- Un stride, que es el ancho en bytes de cada fila de pixeles que incluye el margen interno.

Si todos estos parámetros son correctos al ejecutar el proyecto veremos en pantalla la captura en tiempo real de las imágenes por la cámara del Kinect, por el formato que manejamos en cuadros por segundo podemos asimilar que es una captura de video.

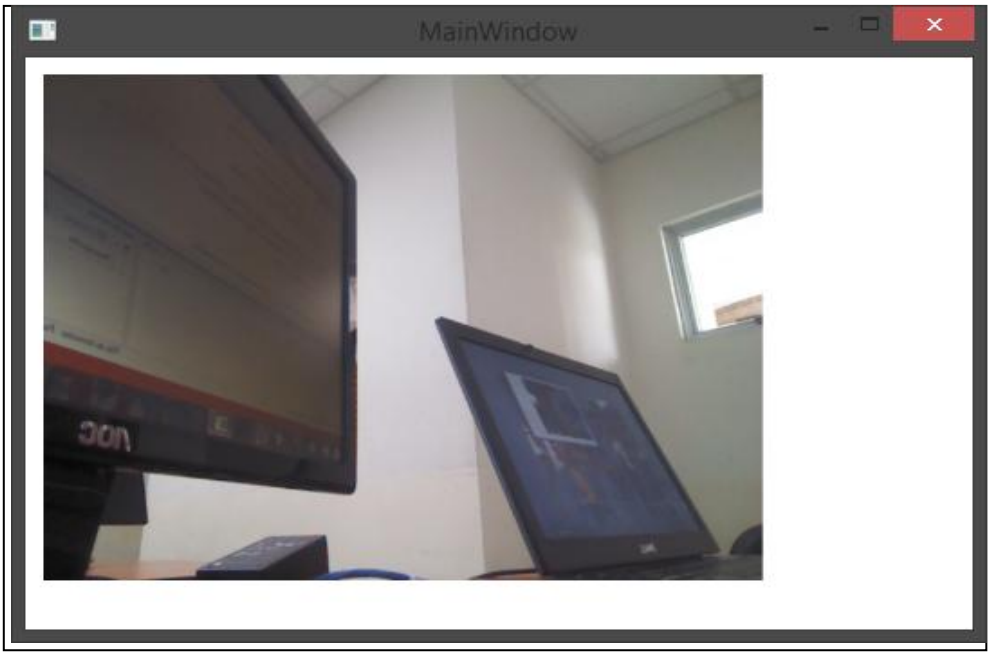


Fig.15 Vista en pantalla de la captura de imágenes por la cámara del Kinect.

6.8 SENSOR DE PROFUNDIDAD.

El sensor de profundidad es un tipo de característica que el Kinect implementa para poder detectar distancias entre los pixeles, lo cual será fundamental para poder calcular distancia e identificar objetos que estén dentro de nuestra escena y generar interacción con los mismos.

Técnicamente hay dos modos de detección de profundidades los cuales son:

- El Near Mode (solo compatible con Kinect for Windows) tiene una capacidad de detencion de distanacias entre 40 cm a 3 m.
- El Default Mode (ambos modelos) tiene el rango de captura entre 80 cm y 4m.

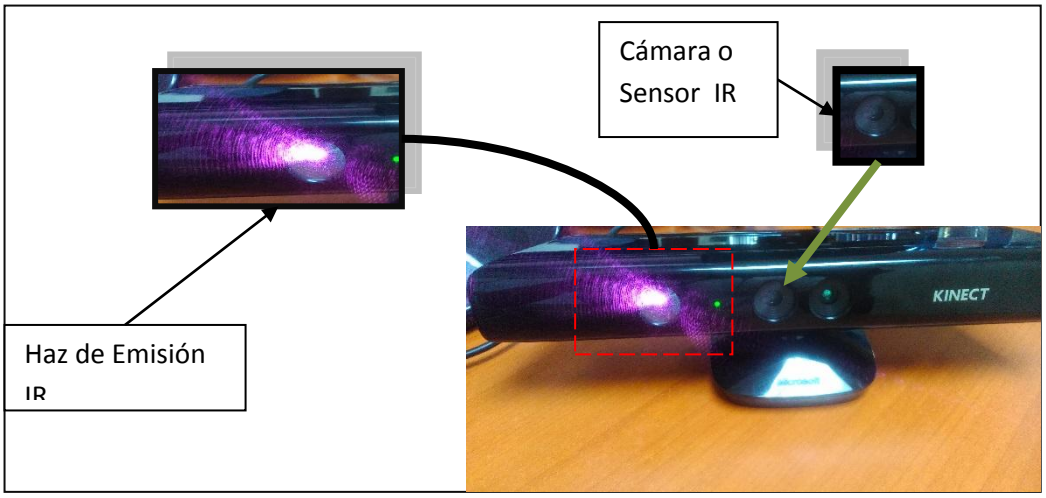
Esta diferencia es significativa si se usa el Kinect para interactuar con equipos de cómputo o consolas de videojuegos ya que en las consolas lo primordial es que exista una separación entre el sensor y la persona capturada y así tener una buena captura de datos.

En cambio en equipos de cómputo se está sentado enfrente de un escritorio y lo común es que se esté a distancias cortas, imagine el escenario que a base de gestos puede pasar de una página web a otra, hacer un dictado y este se guarde en texto directamente en un archivo de texto, en estos escenarios es pensado los diferentes modos de captura de datos de profundidad.

6.9 MANIPULANDO EL SENSOR DE PROFUNDIDAD.

Como se realizó con el sensor de color, realizamos una aplicación que permita capturar y mostrar en pantalla los datos de profundidad y el reconocimiento que esto tienen al combinarlos para formar imágenes.

Primero debemos de saber que el Kinect posee una cámara IR y un proyector de IR, que en el primer proyecto se encendía, el proyector de IR genera un serie de rayos infrarojos que se dispersan en el espacio y chocan contra los objetos, al chocar definen una distancia entre sensor y el objeto con el cual choca.



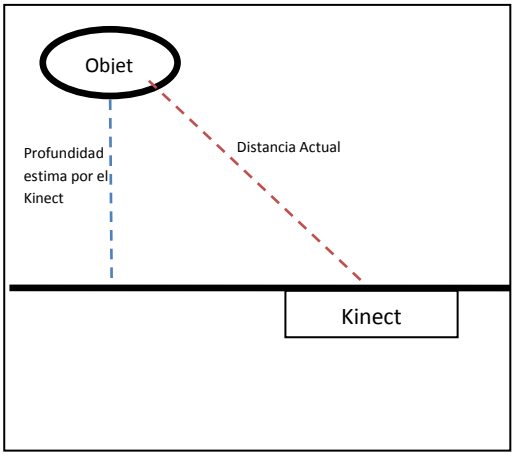
Haz de rayos infrarrojos

Fig.16 Esquema que muestra como la emisión del Kinect toca un objeto y este define una distancia estimada.

Como vemos en la imagen de arriba se puede apreciar el haz de rayos infrarrojos, los cuales se capturan sus rebotes por medio de la cámara infrarroja y con esto se realiza los cálculos para

definir las distancias estimadas, la configuración de captura básica de la cámara tiene los siguientes modos de captura:

- 320 x 240 a 30 fps
- 640 x 480 a 30 fps
- 80 x 60 a 30 fps
- Undefined.



Al saber estas propiedades podemos limitar a una resolución adecuada para los requisitos de nuestro proyecto ahora esta especificación será solamente para limitar el stream de la imagen, los datos que capture la cámara IR siempre se manejaran en un arreglo que lo podemos identificar o analizar con la siguiente gráfica.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D	D	D	D	D	D	D	D	D	D	D	D	D	P	P	P
4096	2048	1024	512	256	128	64	32	16	8	4	2	0	4	2	0

Esta grafica nos define la forma en que se captura el stream de los datos, estos se capturan en un conjunto de 16 bits, pero se debe identificar que solo 13 de los bits capturados corresponden a puntos de profundidad los tres restantes se utilizan para manejar la captura de los esqueletos.

Para manipular las profundidades tenemos que idear una manera de eliminar los datos que no nos sirven para resolver este inconveniente toca hacer uso del operador >> (es un operador de desplazamiento binario, con esto solventaremos el problema.

6.10 CREANDO UNA CAPTURA DE LA CÁMARA DE PROFUNDIDAD.

Para crear esta aplicación solo definiremos el escenario, el cual debe plantearse que capturaremos todos los pixeles que la cámara reconozca pero haremos uso de unos métodos para poder mostrar una vista concreta de acuerdo a la distancia que hay entre el sensor y los pixeles capturados para ello se define la siguiente tabla.

Distancia desconocida	Pixeles de color azul.
Distancia cercana	Pixeles de color Verde
Distancia lejana	Pixeles de color Rojo

Comenzamos a codificar creando los objetos necesarios para poder lograr el objetivo de manipular los datos de la captura de profundidad y mostrarlos en una Contenedor de imagen en el formulario.

Utilizaremos las configuraciones del Ejemplo anterior solo modificamos algunas líneas de código para adaptarse al sensor de profundidad.

Declaramos los objetos del siguiente tipo:

```
KinectSensor mSensor;
short[] datosDistancia = null;
byte[] colorImageDistancia = null;
WriteableBitmap bitmapImagenDistancia = null;
```

En el método ***Window_Loaded*** lo codificamos como se muestra a continuación:

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    if (KinectSensor.KinectSensors.Count == 0)
    {
        MessageBox.Show("No hay Dispositivo Conectado");
        Application.Current.Shutdown();
    }
    try
    {
        mSensor = KinectSensor.KinectSensors[0];
        mSensor.Start();
        mSensor.DepthStream.Enable();
        mSensor.DepthFrameReady += mSensor_DepthFrameReady;
     }
     Catch
     {
        MessageBox.Show("No hay Dispositivo Conectado");
        Application.Current.Shutdown();
    }
}
```

La diferencia significativa en el método es que en lugar de la cámara RGB iniciamos la cámara IR con la instrucción ***DepthStream.Enable(),*** y creamos un manejador del evento para los frames del sensor ,codificamos el método de la siguiente forma.

```
void mSensor_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
{
    using (DepthImageFrame framesDistancia = e.OpenDepthImageFrame())
    {
        if (framesDistancia == null)
        {
            return;
        }
        if (datosDistancia ==null)
        {
             datosDistancia = new short[framesDistancia.PixelDataLength];
        }
        if (colorImageDistancia == null)
        {
            colorImageDistancia = new byte[framesDistancia.PixelDataLength * 4];
        }
        framesDistancia.CopyPixelDataTo(datosDistancia);

        }
     }

}
```

Con este código ya controlamos y se indica dónde y cómo almacenar los datos recibidos por el sensor, ahora debemos manipularlos para ello agregamos seguido de la última línea el siguiente código.

```
int postColorImagenDistancia = 0;
for (int i = 0; i < framesDistancia.PixelDataLength; i++)
{
    //hacemos el desplazamiento de los pixel de esqueleto
    int valorDistancia = datosDistancia[i] >> 3;
    if (valorDistancia == mSensor.DepthStream.UnknownDepth)
    {
        colorImageDistancia[postColorImagenDistancia++] = 0;
        colorImageDistancia[postColorImagenDistancia++] = 0;
        colorImageDistancia[postColorImagenDistancia++] = 255;
     }else if (valorDistancia == mSensor.DepthStream.TooFarDepth)
     {
          colorImageDistancia[postColorImagenDistancia++] = 255;
          colorImageDistancia[postColorImagenDistancia++] = 0;
          colorImageDistancia[postColorImagenDistancia++] = 0;
     }else if (valorDistancia == mSensor.DepthStream.TooNearDepth)
     {
          colorImageDistancia[postColorImagenDistancia++] = 0;
          colorImageDistancia[postColorImagenDistancia++] = 255;
          colorImageDistancia[postColorImagenDistancia++] = 0;
     }
     else
     {
         byte byteDistancia = (byte)(255 - (valorDistancia >> 2));
         colorImageDistancia[postColorImagenDistancia++] = byteDistancia;
         colorImageDistancia[postColorImagenDistancia++] = byteDistancia;
         colorImageDistancia[postColorImagenDistancia++] = byteDistancia;
     }
     postColorImagenDistancia++;
}
```

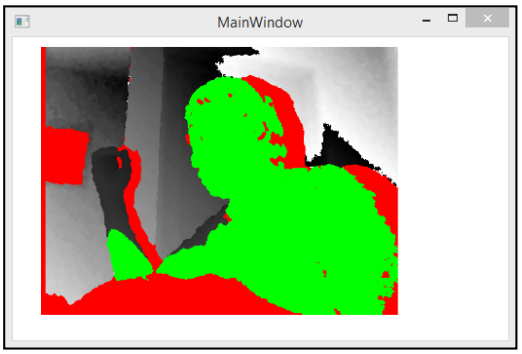
Con estas instrucciones aplicaremos los parámetros que definimos al inicio, donde se define como colorear cada pixel dependiendo de la distancia.

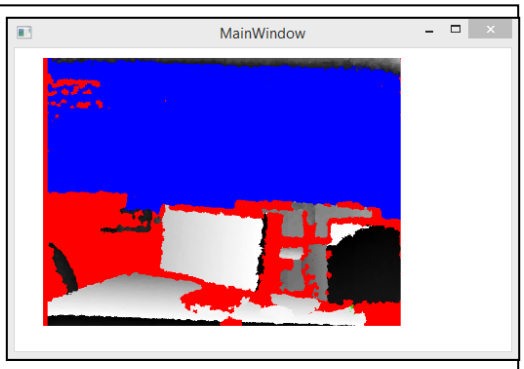
Solo resta enviar los datos al control de la imagen para poder visualizar los resultados, para ello agregamos las instrucciones siguientes.

```
if (bitmapImagenDistancia == null)
{
    this.bitmapImagenDistancia = new WriteableBitmap(
    framesDistancia.Width,framesDistancia.Height,
    96,96,
    PixelFormats.Bgr32
    null);

     videoSource.Source = bitmapImagenDistancia;
 }

 this.bitmapImagenDistancia.WritePixels(
     new Int32Rect(0,0,framesDistancia.Width,framesDistancia.Height),
     colorImageDistancia,
     framesDistancia.Width * 4,
     0
     );
```

Con esto ya tenemos creado el método ahora queda hacer la prueba, solo hay que advertir un cambio en la forma de manipular el objeto donde se construye la imagen ya que en este caso se

ha utilizado un ***WritableBitmap***, esto es para optimizar el uso de memoria.





Resultado de la ejecución del código, podemos observar los distintos colores, los cuales indican las distancias estimada cada pixel.

6.11 SENSOR DE ESQUELETOS.

La última característica que identificaremos en la parte de captura de datos por las cámaras es la definición de los puntos de esqueleto.

Esta es la principal característica por la cual el Kinect se ha convertido en un dispositivo con grandes características de aplicación para ser aprovechadas a la realidad aumentada, y eso lo podemos observar en los juegos desarrollados para la consola Xbox 360 de Microsoft ya que el jugador es detectado para poder ser parte del juego y este ya no utiliza ningún control adicional más que su cuerpo.

Para lograr tal interacción los desarrolladores del sensor crearon rutinas que detectan 22 puntos específicos del cuerpo de una persona estando de pie, estos puntos se reconocen como parte de una especificación que denominaremos puntos de esqueleto, los cuales al detectarse, combinarse y estar en diferentes ángulos y distancias entre ellos, indican las acciones que en el caso de los juegos, disparan eventos para que se cumplan los objetivos del juego.

Kinect Sports

Just Dance 3

NBA Beat Ballers

Nike+ Kinect Training

Fig. 17 Algunos Juegos que aprovechan las características de detección de Kinect.

La profundidad, las imágenes a colores son grandes cualidades a la hora de hacer realidad aumentada, pero en Kinect para lograr una interacción grande entre los entornos se destaca la característica de la detección de los esqueletos sin esto, Kinect solo sería un proyecto de captura de imágenes y comparación, sin mayor interacción entre el mundo real y el virtual.

6.12 LA TEORÍA DE LOS PUNTOS.

Como se ha logrado investigar la detección del Kinect especifica 20 puntos del cuerpo de pie, hay un método diferente de detección, el cual es el modo seat(solo Kinect for Windows) que solo detecta 10 puntos identificamos los puntos de acuerdo a su posición relativa a una parte del cuerpo, para identificar los puntos veamos la siguiente gráfica.

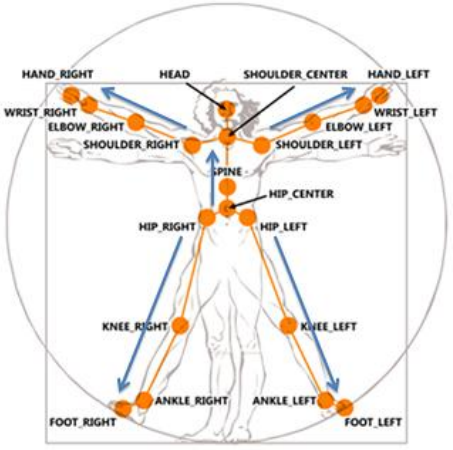


Fig. 18Puntos detección Kinect

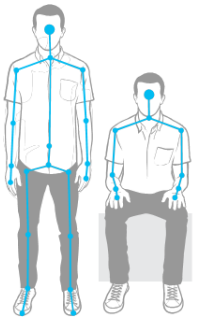


Fig. 19 Puntos según Modo Default/Seated

Identificando los puntos podemos ver que están distribuidos a lo largo de las partes del cuerpo humano y casi todos están en pares de izquierda y derecha.

Los puntos al ser detectados, en un brazo por ejemplo, siempre realizaran un seguimiento del brazo al moverlo, estos son parte de los algoritmos de detección que posee el Kinect, porque hay que mencionar esto es porque al hacer un seguimiento de la parte del cuerpo y el punto en específico podemos detectar la posición y comprobar haciendo un método de triangulación para disparar una acción por ejemplo:

Si el punto del codo está alineado a 90 grados con el punto del hombro y el punto de la muñeca podemos decir que el sujeto **a** tiene su mano levantada y este es un gesto el cual puede ser reconocido y detectado para realizar una acción **x** en nuestro programa.

Otra de las características del sensor y de la detección de los puntos es que este puede identificar su posición en tres dimensiones y con esto se puede ampliar la cantidad de gestos y movimientos para poder realizar acciones en nuestras aplicaciones.

Un detalle técnico que se toma en cuenta es que la detección de cuerpos o esqueletos se limita a dos esqueletos completos o (la serie de puntos), esta limitación es del sensor, así que al desarrollar nuestras aplicaciones podemos crear un sistema de detección únicamente para dos usuarios a la vez.

En la detección de los esqueletos el sensor analiza en qué posición se encuentra ya que de esto depende si está en una buena área de captura y obtener datos más fiables, pero si no es el caso, hay ciertos métodos que están implementados en la API para mejorar estas características y hacer que el usuario se coloque en una posición adecuada para obtener los datos óptimos de la captura de su esqueleto.

Para manipular de una forma adecuada esta situación nos basamos en los estados de detección del sensor, de acuerdo a este estado podemos realizar las rutinas necesarias para mejorar la calidad de información, los estados son los siguientes:

- **Tracked**, es el estado normal el dato está en la zona adecuada de captura.
- **Inferred**, en este estado se da al asumir la posición del punto en base a otro punto ya sea porque alguno de ellos está enfrente u obstaculiza la zona de captura.
- **NotTracked**, Kinect no puede determinar la posición del punto.

6.13 MANIPULANDO EL SKELETON STREAM

Creamos una simple aplicación que nos diga las coordenadas de un punto. Como en los anteriores ejemplos usaremos la misma base para el proyecto, la diferencia básica para este será que en el objeto Window_Loaded activaremos el **Skeleton Stream** y definiremos un método para controlar los datos obtenidos del Kinect. El siguiente código muestra la manera de implementar estas acciones:

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
         if (KinectSensor.KinectSensors.Count == 0)
         {
             MessageBox.Show("Ningun dispositivo encontrado");
             Application.Current.Shutdown();
             return;
          }
          mSensor = KinectSensor.KinectSensors[0];
          try
          {
              mSensor.SkeletonStream.Enable();
              mSensor.Start();
              mSensor.SkeletonFrameReady += mSensor_SkeletonFrameReady;
          }
          catch
          {
               MessageBox.Show("Ningun dispositivo encontrado");
               Application.Current.Shutdown();
          }
  }
```

Creando el Event Handler:

```
void mSensor_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
    string mensaje = "No hay datos de esqueleto";
    string mensajeCalidad = "";
    Skeleton[] esqueletos = null;
    using (SkeletonFrame framesEsqueletos = e.OpenSkeletonFrame())
    {
        if (framesEsqueletos != null)
        {
            esqueletos = new Skeleton[framesEsqueletos.SkeletonArrayLength];
            framesEsqueletos.CopySkeletonDataTo(esqueletos);
         }
     }

     if (esqueletos == null)
     {
         return;
     }

}
```

Con estas líneas de código preparamos el objeto ***Skeleton***, el cual maneja los datos obtenidos desde el sensor, los datos que obtiene son almacenados en el arreglo de tipo ***Skeleton*** que ese declara dentro del método, para poder manipular y hacer las rutinas necesarias para las acciones que deseamos con nuestro programas, en el ejemplo complementaremos el método con el siguiente código:

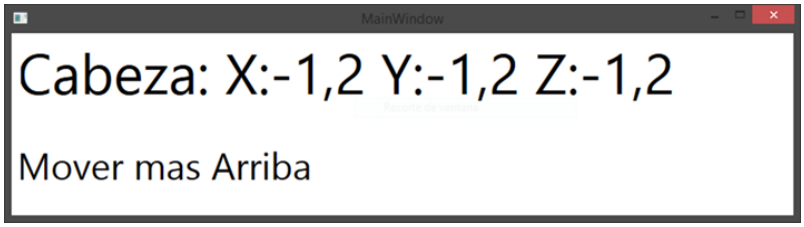
```
foreach(Skeleton esqueleto in esqueletos)
{
    if (esqueleto.TrackingState == SkeletonTrackingState.Tracked)
    {
        if (esqueleto.ClippedEdges == 0)
        {
            mensajeCalidad = "En Posicion";
        }
        else
        {
            if ((esqueleto.ClippedEdges & FrameEdges.Bottom) !=0)
            {
                mensajeCalidad += "Mover mas Arriba";
            }
            if ((esqueleto.ClippedEdges & FrameEdges.Top) != 0)
            {
                mensajeCalidad += "Mover mas Abajo";
            }
            if ((esqueleto.ClippedEdges & FrameEdges.Right) != 0)
            {
                mensajeCalidad += "Mover mas a la Izquierda";
            }
            if ((esqueleto.ClippedEdges & FrameEdges.Left) != 0)
            {
                mensajeCalidad += "Mover mas a la Derecha";
            }
        }

        Joint jointCabeza = esqueleto.Joints[JointType.Head];
        SkeletonPoint posicionCabeza = jointCabeza.Position;
        mensaje = string.Format("Cabeza: X:{0:0.0} Y:{0:0.0} Z:{0:0.0}",
                    posicionCabeza.X, posicionCabeza.Y, posicionCabeza.Z);
        textBlockEstatus.Text = mensaje;
        textBlockCaptura.Text = mensajeCalidad;
        }
 }
```

El código define las siguientes acciones para poder obtener el resultado especificado:

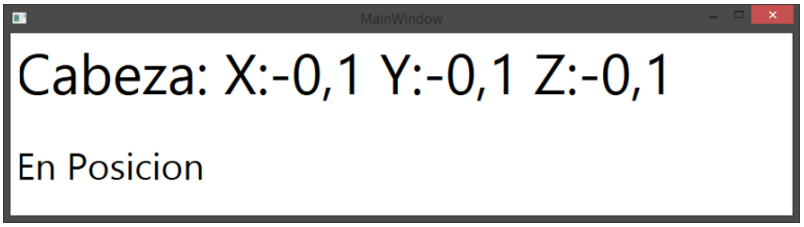
1. Se recorre el arreglo que almacena los puntos (Joints) del esqueleto
2. Se verifica la posición del esqueleto si está en la zona óptima de captura, si esta lo indicamos con un mensaje, en caso contrario el mensaje indica o define que debemos hacer para estar en la zona óptima.
3. Trabajamos con un simple punto para saber su posición y mostrarla en pantalla.

4. Obtenemos las posiciones X,Y,Z y las mostramos en un texto de la pantalla.

Con el resultado es ver los datos de posición en pantalla como se muestra en las imágenes.



Datos obtenidos de la posición de la cabeza, y advertencia de que la captura no es óptima



Datos obtenidos de la posición de la cabeza, en posición óptima.

El resultado verifica específicamente el punto de la cabeza este puede ser cambiado usando las definiciones predeterminadas que nos brinda la API de Kinect, las identificaciones de cada punto se muestran en la siguiente tabla

Identificador	Descripción
AnkleLeft	Talón Izquierdo
AnkleRight	Talón Derecho
ElbowLeft	Codo Izquierdo
ElbowRight	Codo Derecho
FootLeft	Pie Izquierdo
FootRight	Pie Derecho
HandLeft	Mano Izquierda
HandRight	Mano Derecha
Head	Cabeza
HipCenter	Centro de la Cadera
HipLeft	Cadera Izquierda

HipRight	Cadera Derecha
KneeLeft	Rodilla Izquierda
KneeRight	Rodilla Derecha
ShoulderLeft	Hombro Izquierdo
ShoulderRight	Hombro Derecho
Spine	Base de la columna
ShoulderCenter	Centro de los Hombros
WristLeft	Muñeca Izquierda
WristRight	Muñeca Derecha

Identificado estos puntos podemos crear una aplicación en la cual se puede dibujar las 19 uniones o huesos que componen los esqueletos que detecta el Kinect para ello debemos de contar con objeto en el cual podamos dibujar en tiempo real el objeto recomendado es el ***canvas,*** ya que es como tener un lienzo en el cual podemos dibujar las formas deseadas.

Lo que nos ayuda a crear estas líneas es que tenemos un punto de inicio y un punto final, entonces haciendo uso de las API de dibujo se puede implementar, un código siguiendo la siguiente lógica.

- Definir los puntos que queramos que sean el inicio y el fin.
- Definir el objeto que se dibuja.
- Identificar posición x e y del punto de inicio de la línea a dibujar y asociarlo al punto 1 detectado por el kinect
- Identificar posición x e y del punto fin de la línea a dibujar y asociarlo al punto 2 detectado por el kinect.

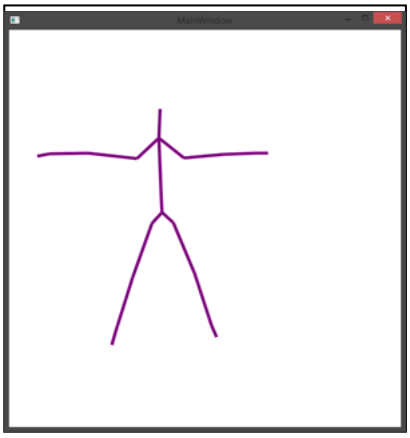
Solo hay que saber que los puntos detectados son datos de ubicación espacial, para poder dibujarse en el escenario se deben de convertir a puntos de color y que este coordinados en el mapeo ya que, si esto no se realiza, tendremos en pantalla una línea que no coincide con el cuerpo del usuario detectado o se mostrara con un desfase en el movimiento.

Para hacer la conversión Utilizaremos la función:
CoordinateMapper.MapSkeletonPointToColorPoint()

Pequeño trozo de código que muestra el dibujo de una línea usando los puntos de detección del Kinect:

```
Joint cabezaPunto = esquelto.Joints[JointType.Head];
Joint hombroCentroPunto = esquelto.Joints[JointType.ShoulderCenter];


ColorImagePoint puntoCabeza =
mSensor.CoordinateMapper.MapSkeletonPointToColorPoint(cabezaPunto.Position,
ColorImageFormat.RgbResolution640x480Fps30);
                    ColorImagePoint puntoHombroCentro =
mSensor.CoordinateMapper.MapSkeletonPointToColorPoint(hombroCentroPunto.Position,
ColorImageFormat.RgbResolution640x480Fps30);

Line linea = new Line();

linea.Stroke = new SolidColorBrush(Colors.Purple);
linea.StrokeThickness = 5;

linea.X1 = puntoCabeza.X;
linea.Y1 = puntoCabeza.Y;
linea.X2 = puntoHombroCentro.X;
linea.Y2 = puntoHombroCentro.Y;
```

Esto se debe hacer por cada hueso que se dibuje en el escenario, al dibujar los 19 huesos tenemos una salida como la siguiente:



7 CONCLUSIONES

Conociendo la forma de poder identificar los puntos, podemos crear interacciones complejas, crear un entorno de Realidad aumentada mucho más dinámico que usar solo una captura de un patrón, o el uso de otra técnica, haciendo uso de otras herramientas podemos crear objetos 3D para interactuar con ellos solo debemos crear las interfaces que permitan, la interacción de los puntos de esqueleto y dichos objetos.

Blender es un proyecto open source con el podemos crear objetos 3D, tiene grandes ventajas a otros programas libres, y llega a las posibilidades de modelado de programas comerciales como el 3D Max de Autodesk y Maya, con este podemos crear además de objetos escenarios para interactuar junto con los objetos.

Para dar más posibilidades de crear realidad aumentada se debe combinar algunas técnicas como las colisiones de objetos, el uso de objetos 2D y 3D, definición de objetivos y gestos que disparen o desencadenen eventos, esto creado desde cero nos tomaría mucho tiempo, llegando a resultados que pueden limitar nuestras expectativas, es por ello que es recomendable que podamos usar algún software que facilite dichas tares y es por ello que los Motores de desarrollo de videojuegos se adaptan de una manera muy óptima para estos objetivos.

Podemos aprovechar específicamente Unity 3D que es un motor libre y que entre su comunidad de desarrollo hay un proyecto el cual permite usar de forma nativa las capturas del Kinect para poder generar las interacciones en los mundos 2D o 3D que se desarrollen.

Combinando estos componentes podemos adentrarnos al desarrollo de escenarios de Realidad Aumentada que se puedan aprovechar no solo en la diversión, sino en el entrenamiento y educación de las personas.

8 RECOMENDACIONES

Se considera necesario:

- Profundizar en el manejo de las imágenes que permite el KinectFor Windows, esto para formular un diseño adecuado a dicha tecnología.
- Estudiar el manejo de las API de detección de movimiento y rastreo de esqueleto.
- Aplicar uso del reconocimiento de voz.
- Crear prototipos para las pruebas del conjunto de los componentes para la realización de la plataforma.

9 BIBLIOGRAFÍA

- Catuhe, David. Programming with the Kinect for Windows Software Development Kit. Redmond, Wash: Microsoft Press, 2012.

- Giori, Clemente, and Massimo Fascinari. Kinect in motion audio and visual tracking by example. Birmingham, UK: Packt Pub, 2013.

- Miles, Rob S. Start here! Learn Microsoft Kinect API. Sebastopol, Calif: O'Reilly Media, 2012.

- Jana, Abhijit. Kinect for Windows SDK Programming Guide. Birmingham: Packt Pub, 2012.

- José M. Aguilar. Usando using, valga la redundancia (C#), 2009 http://www.variablenotfound.com/2009/02/usando-using-valga-la-redundancia-c.html, Septiembre 2014.
- Microsoft Developer Network. ColorImageFrame.CopypixelDataTo Method https://msdn.microsoft.com/en-us/library/microsoft.kinect.colorimageframe.copypixeldatato.aspx, Septiembre 2014
- Microsoft Developer Network. BitmapSource.Create Method (Int32, Int32, Double, Double, PixelFormat, BitmapPalette, Array, Int32) https://msdn.microsoft.com/en-us/library/ms616045(v=vs.110).aspx, Octubre 2014
- Microsoft Developer Network. Coordinate Space, https://msdn.microsoft.com/en-us/library/hh973078.aspx, enero 2015.
- Tom Kerkhove. Introduction to Kinect for Windows SDK(01-04-2013) http://www.kinectingforwindows.com/2013/04/01/introduction-to-kinect-for-windows-sdk/, enero 2015
- Microsoft Developer Network, JointType Enumeration,https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx, Enero 2015.

UN FUTURO LLENO DE OPORTUNIDADES

Escuela Especializada en Ingeniería

ITCA FEPADE

SANTA TECLA · ZACATECOLUCA · SAN MIGUEL · SANTA ANA · LA UNIÓN

megatec

EDUCACIÓN TÉCNICA, TECNOLÓGICA Y SUPERIOR

MINISTERIO DE EDUCACIÓN

GOBIERNO DE EL SALVADOR

UNÁMONOS PARA CRECER

Sede Central Santa Tecla

Km. 11 Carretera a Santa Tecla.

Tel. (503) 2132-7400

Fax. (503) 2132-7599

Centro Regional

MEGATEC La Unión

C. Santa María, Col. Belén, atrás del

Instituto Nacional de La Unión.

Tel. (503) 2668-4700

Centro Regional

MEGATEC Zacatecoluca

Km. 64 1/2, desvío Hacienda El Nilo, sobre autopista a Zacatecoluca y Usulután. Tel. (503) 2334-0763, (503) 2334-0768 Fax. (503) 2334-0462

Centro Regional San Miguel

Km. 140, Carretera a Santa Rosa de Lima.

Tel. (503) 2669-2292, (503) 2669-2299
Fax. (503) 2669-0961

Centro Regional Santa Ana

Final 10a. Av. Sur, Finca Procavia

Tel. (503) 2440-4348, (503) 2440-2007

Tel. Fax. (503) 2440-3183

Escuela Especializada en Ingeniería

ITCA FEPADE