SISTEMA DE CAPTURA DE MOVIMIENTO PARA ANIMACIÓN DE PERSONAJES 3D Y PRESENTACIONES DE ALTO IMPACTO

Autor
WLADIMIR GARCES CARRILLO

Director
CÉSAR AUGUSTO PEÑA
PhD. Automática y Robótica

INGENIERÍA MECATRÓNICA
DEPARTAMENTO DE MECÁNICA, MECATRÓNICA E INDUSTRIAL
FACULTAD DE INGENIERÍAS Y ARQUITECTURA

UNIVERSIDAD DE PAMPLONA
2017

SISTEMA DE CAPTURA DE MOVIMIENTO PARA ANIMACIÓN DE PERSONAJES 3D Y PRESENTACIONES DE ALTO IMPACTO

Autor
WLADIMIR GARCES CARRILLO

Director
CÉSAR AUGUSTO PEÑA
PhD. Automática y Robótica

INGENIERÍA MECATRÓNICA
DEPARTAMENTO DE MECÁNICA, MECATRÓNICA E INDUSTRIAL
FACULTAD DE INGENIERÍAS Y ARQUITECTURA

UNIVERSIDAD DE PAMPLONA
PAMPLONA, 2017

DEDICATORIA

Este libro va dedicado a mis padres Margot Carrillo Torreglosa y Néstor Raúl Garcés Quintero, a mis abuelos Virgilio Carrillo y Tilda Torreglosa, quienes me apoyaron en este proceso que aún no termina, y cuya meta es la de trascender en este planeta y quienes con sus consejos me han guiado por el camino. A mi hermano, hermanas y sobrinos a quienes les estoy preparando un futuro mejor. Se lo dedico a mis profesores, que, con escepticismo, han puesto un grano de confianza en mí.

Espero que la información que está contenida aquí sirva para alimentar trabajos futuros de compañeros que busquen nuevas posibilidades.

AGRADECIMIENTO

Agradezco a cada una de las personas que han confiado en mis capacidades y que, con sus consejos, buenos deseos y ánimo, me han hecho pensar siempre en salir adelante. A los dedos pulgares que nos dieron la capacidad del uso de herramientas, a la evolución del cerebro como órgano conductor de nuestras decisiones.

Agradezco al estudio del espacio, que ha dado grandes avances a la tecnología y el descubrimiento de nuevos materiales para que podamos tener mejores capacidades para el desarrollo continuo de la humanidad.

Agradezco a mi condición latinoamericana, si no hubiese nacido en este país, no tuviese la configuración de pensamiento que hoy día me aborda y con el que pretendo la construcción de un nuevo escenario, para que mi pueblo se desarrolle y se cree mejores metas, para así, poder romper todos los paradigmas que nos amarran a soluciones obsoletas y sin sentido alguno y lleguemos por fin a ser la gran nación que somos.

RESUMEN

La propuesta consiste en la creación de un sistema de captura de movimiento (MOCAP), para el modelado y animación de la mascota de la universidad, con el fin de generar material audiovisual educativo para la promoción institucional y para la creación de presentaciones de alto impacto.

En su desarrollo fue necesario la creación, por medio de modelado grafico en 3D, de una de las mascotas de la universidad, y practicar sobre ella técnicas de animación y de modificación morfológica de “mallas”, en este caso la de *rigging*. Fueron necesarios conceptos en animación combinados con conceptos en robótica para la creación completa del rigging del personaje.

También se necesitó para su desarrollo un análisis minucioso del sensor Kinect y de la librería PyKinect, usada para la extracción específica del *Skeleton Tracking* para su posterior implementación en la creación de gestos a partir del seguimiento de los puntos de las muñecas y para la extracción de los ángulos para el control del rigging del personaje hecho en Blender. Cabe especificar que los softwares se escribieron, enteramente en Python con ayuda de otras librerías como PyQt4 para el desarrollo de la interfaz, OpenCV para la visualización de los datos obtenidos o PyWin32 para emular eventos de teclado y mouse.

Para la realización de los software’s, fueron necesarios conceptos como Programación Orientada a Objetos, programación Orientada a Eventos y Programación en Paralelo por medio de la ejecución de hilos. También se usó una lógica de control I/O para algunas órdenes de control de distintas etapas del programa.

Teniendo en cuenta todo lo anterior. Se desarrolló un sistema de captura de movimiento capaz de recolectar los movimientos de una persona y disponerlos en un archivo de datos ordenados, para que posteriormente sea leído por otro programa para su reproducción en Blender.

Se desarrolló un *Script* en Python, para ejecución en Blender, que permite cargar un archivo de datos ordenados, leerlo y traducir cada línea a movimiento de un robot virtual.

Se desarrolló un personaje en 3D, a partir de la imagen institucional de la Universidad de Pamplona. Esta imagen podría servir para representar la universidad en sus distintas iniciativas para la recolección, reproducción, creación y desarrollo de material virtual de apoyo para los distintos materiales virtuales de la institución y su posterior implementación en las Aulas TIC.

Se logró el desarrollo un sistema de captura de movimiento, como herramienta tecnológica y con propósitos de uso en las Aulas TIC, para el desarrollo de presentaciones de alto impacto. Esta herramienta permite controlar una diapositiva por medio de gestos de las extremidades superiores. Este proyecto entra como insumo a las propuestas a nivel mundial que quieren potencializar la educación por medio de herramientas tecnológicas que sirvan

para reproducir el conocimiento de manera efectiva a las nuevas generaciones de la sociedad tecnológica.

Por último, los resultados esperados, como se dijo al principio, se verán reflejados en: La ***animación de las mascotas***, que representan hoy, la ***imagen corporativa de la Universidad de Pamplona***, y que la dotarán con producción audiovisual de calidad para el fortalecimiento de su gestión publicitaria; y en la ***gestión de presentaciones por medio de gestos corporales***, que se enmarquen en el actual contexto de las ***presentaciones de alto impacto***. A futuro, se cree que este sistema podrá tener otras aplicaciones como: Sistemas de rehabilitación deportiva, videojuegos, etc.

Palabras Claves:

MOCAP, fotogrametría, procesamiento de imágenes, captura de movimiento, presentaciones de alto impacto, Animación, rigging, Skeleton Traking, Paradigma de Programación.

CONTENIDO

LISTA DE FIGURAS

CAPITULO 1.
SISTEMA DE CAPTURA DE MOVIMIENTO PARA ANIMACIÓN DE PERSONAJES 3D Y PRESENTACIONES DE ALTO IMPACTO

1.1. OBJETIVOS

1.1.1. Objetivo General

Diseñar e implementar un sistema de captura de movimiento para animación de personajes en 3D y presentaciones de alto impacto.

1.1.2. Objetivos Específicos

- Modelado gráfico en 3D de 2 mascotas de la Universidad.
- Diseñar un algoritmo capaz de capturar y registrar las relaciones corporales del movimiento de las extremidades.
- Desarrollar un algoritmo capaz de reproducir movimiento de personas en los modelos en 3D de las mascotas de la universidad previamente diseñados teniendo en cuenta los registros antes obtenidos.
- Animar las 2 mascotas de la universidad.
- Diseñar un algoritmo que permita el control de presentaciones mediante gestos corporales.

1.2. PROBLEMA

La Universidad de Pamplona encamina su búsqueda por el posicionamiento de calidad en los escalafones correspondientes en el país respecto a implementación de una política TIC que sea aplicable a los procesos educativos desde la creación, construcción y desarrollos de metodologías, estrategias o algún enfoque que conlleve a la inclusión de las TIC´s en las vidas cotidianas de sus educadores, también mediante el apoyo de proyectos tecnológicos o educativos con estos mismos fines a través de su dependencia PLANESTIC. Esta dependencia impulsa la creación de escenarios propicios para el desarrollo de las TIC´s como lo es, las AULAS TIC´s. También es la encargada de la vigilancia de la producción de material virtual para el enriquecimiento de estos escenarios. Así que, apoyan la construcción de una herramienta adicional para el apoyo didáctico en dichos espacios que los dote con mayor complejidad, como, por ejemplo, las técnicas de animación 3D para la producción del material virtual antes mencionado, o la creación de los módulos de las materias virtuales.

Un problema más se encuentra en la promoción de los programas académicos y de las actividades curriculares y extracurriculares de la Universidad de Pamplona y de la universidad misma, a la cual se le podría incluir una actualización de sus herramientas de desarrollo y una mejora en su imagen. Para ello las técnicas avanzadas serian de gran utilidad.

En la actualidad las herramientas virtuales van ganando más importancia, debido al gran desarrollo que se ha tenido en la tecnología. Las técnicas de modelado y animación 3D, son parte fundamental para la producción de piezas virtuales con características específicas, gran ejemplo de esto se puede notar en la producción cinematográfica o en la industria de los videojuegos. También se pueden utilizar estas herramientas en publicidad y en la producción de materiales didácticos. En general, en la industria del entretenimiento tiene gran campo este tipo de alternativas virtuales. Sin embargo, se ha abierto camino para un sinnúmero de aplicaciones y se ha extendido, casi a la vida cotidiana, ejemplo claro de esto es la cantidad de herramientas virtuales que se utilizan en las aplicaciones web y en aplicaciones móviles.

1.3. JUSTIFICACIÓN

Los sistemas de captura de movimiento hoy día toman más auge debido a que facilitan el estudio del mismo en distintas áreas. Sin embargo, quienes se han podido ver más beneficiados es la industria del video juego y el cine, siendo estos los que han brindado sus mayores avances.

Sin embargo, en estas últimas décadas la introducción de las Tecnologías de la Informática y Comunicaciones sugiere que cada vez se creen mejores contenidos informáticos en perspectivas de la educación, por ende, en países del nuevo y viejo continente se nota la preocupación en este campo desde el escenario político. La neurociencia ha dicho que los profesores deben reeducarse, debido a que los alumnos hoy por hoy tienen acceso a mucha más información, obligando que la relación de los maestros y estudiantes vaya perdiendo interés. Esta relación que por ende no se puede obviar, debido a que es “La interacción mutua de dos cerebros, donde intervienen emociones, sensaciones y en algún aspecto, racionalidad” (Dr. Manes, 2015). En este sentido, son muchas las propuestas que se tienen entorno a la potencialización de la reproducción efectiva de la información, de calidad, confiable y de alta efectividad. Como el ejemplo de Pere Marqués con su “Pizarra digital”. (Marquès Graells, 2006)

Los gobiernos también han puesto su atención a esta problemática, reconociendo la sociedad de la información y el conocimiento como un escenario que cada vez gana más espacio en el mundo y que sin el desarrollo de estas no se podrá crecer en un

mundo que acelera cada vez más su paso en la producción de tecnología. La ley 1341 motiva a las entidades educativas a que aplique esta clase de conocimiento en sus actividades diarias, así también, las exhorta a la creación de material tecnológico, académico o intelectual, de algún tipo de herramientas que sirvan a las TIC (colombia, 2009). En este sentido la Universidad de Pamplona por medio de PLANESTIC, incentivando por el complimiento de la ley colombiana, fomenta la creación de nuevas alternativas que sirvan para el uso de nuevas tecnologías en la educación. Por esto se crea y se ubican distintas necesidades como la de potenciar las Aulas-TIC construidas al interior del plantel educativo, con la creación de herramientas que ayuden a su fortalecimiento y avance.

1.4. SOLUCIÓN

Para darle solución al problema planteado se diseña y crea un sistema de captura de movimiento implementando Kinect, que por un lado sea capaz de reportar gestos para el control de presentaciones, descritas como de alto impacto, debido a su implementación junto a una metodología descrita en este mismo documento. Por otro lado, un software que capture los movimientos de una persona mediante las relaciones angulares de su cuerpo, para su posterior reproducción en un personaje virtual que en otras denominaciones podría ser descrito como un *"Robot virtual"*.

CAPITULO 2.
MARCO TEÓRICO Y ESTADO DEL ARTE

2.1. CAPTURA DE MOVIMIENTO (MOCAP)

Nota: En esta sección se determina la abreviación MoCap, como la adecuada para referirnos al término “Captura de Movimiento”.

2.1.1. Historia de MoCap

Precursores.

La captura de movimiento tiene su inicio con las primeras personas que analizaron el movimiento en la antigua Grecia. ***Aristóteles,*** en su obra *“Motu Animalium”* describe y compara los cuerpos de los animales con sistemas mecánicos de la época. ***Leonardo da Vinci,*** con casi 2 mil años de diferencia describió algunos de los mecanismos que utiliza el cuerpo humano para moverse: Como mantenerse de pie, como saltamos, nos levantamos tras estar sentados, como caminamos en pendientes, etc. ***Galileo*** 100 años más tarde tratará de analizar la fisiología humana desde el punto de vista matemático. ***Borelli,*** con ayuda de Galileo, calculó las fuerzas necesarias para mantener el equilibrio en varias articulaciones del cuerpo, determinó el centro de gravedad del ser humano, así como los volúmenes de aire que el ser humano Inspiraba y Expiraba, demostrando que la respiración era debida a la elasticidad de los tejidos. (Kitagawa & Windsor, 2008)

Época moderna: Nacimiento de la captura de movimiento.

La época moderna fue inspiradora y produjo los avances más grandiosos, uno de ellos fue la *fotografía.* La fotografía permitió que se pudiera capturar una posición determinada del cuerpo para que fuera mejor su análisis.

En 1872 ***Eadweard Muybridge*** (1830-1904), financiado por *Leland Stanford*, es capaz de descomponer el movimiento humano y animal, nace la idea de tomar el movimiento de un caballo, con múltiples cámaras dispuestas hacia el objetivo, se toman fotografías secuenciales.

Seis años después, en 1879 *Muybridge* inventa el ***Zoopraxiscopio.*** Considerado el primer sistema de captura de movimiento, disponía una plataforma circular en cuyo alrededor se ubicaban cámaras apuntando a un objetivo, este tomaba imágenes secuenciales. Muybridge documentará los resultados en sus obras *Animals in Motion (1899) y The Human Figures in Motion (1901),* en estas se encuentran las imágenes

de atletas, niños, deportistas y animales, tomadas con este sistema. (Menache, 2010) (Gutiérrez Lira, 2006).

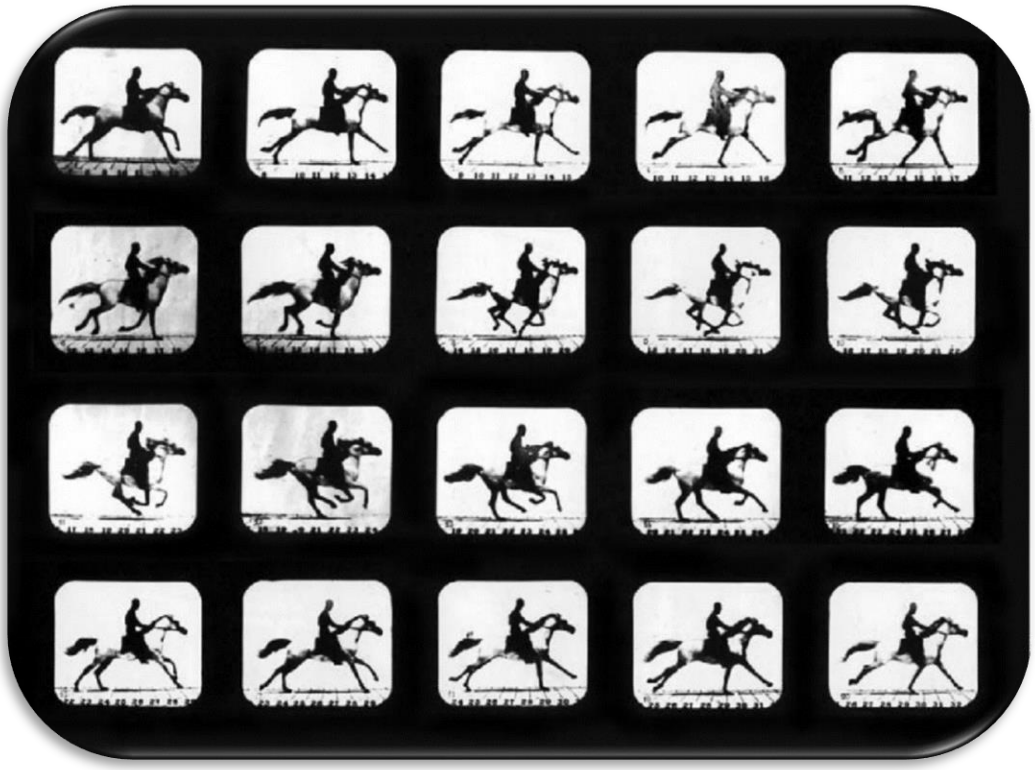


Figura 2 1 Mahomet Running, Eadweard Muybridge, 1879. (Kitagawa & Windsor, 2008).

Las técnicas de Muybridge fueron utilizadas también para el diseño de dibujos animados por empresas de la animación como *Walt Disney*, que lo utilizó para animar sus producciones, la primera fue *Blanca Nieves y los Siete Enanitos.* La animación de personajes no sería la misma desde ese punto.

A finales de los *70´s* y principio de los *80´s*, la animación por ordenador tuvo auge, hasta el punto el que una vez tomado el movimiento se diseñaba sobre esas imágenes, como la técnica de calcado. También fue común en este período que las técnicas de captura se usaran en estudios de *Biomecánica*. El profesor de Kinesiología y ciencias de la computación de la Universidad de Simón Fraser, ***Tom Carvert***, fue capaz de adherir potenciómetros al cuerpo de una persona para animar personajes animados por computadora, con el fin de usarlos en áreas específicas de la danza y la medicina.

El *"traje"* de Carvert era como un exoesqueleto con los potenciómetros dispuestos en las articulaciones de manera que la salida de los potenciómetros mostrara el grado

de flexión de las articulaciones. Una vez digitalizada, la señal seria usada para simular los movimientos en la computadora. Esta técnica, conocida como ***MoCap por accionamiento electromecánico*** o simplemente ***MoCap Mecánico,*** será descrita más adelante, variaciones y configuraciones más complejas serian usados también en distintas formas animadas tales como *Waldo de Pacific Data Images* (***Figura 2.2***) (Kitagawa & Windsor, 2008).

Figura 2 2 (a) El aparato de captura y (b) Waldo C. Graphics.

2.1.2. Tipos de MoCap

Por lo visto anteriormente, el precursor de las formas de capturar movimiento ha sido el ***Método Óptico de MoCap*** con el surgimiento de la fotografía, sin embargo, la reproducción de movimiento como tal en la *"era digital",* se dio a través de la captura formulada por el Kinesiólogo Tom Carvert, por medio de un exoesqueleto con potenciómetros en las articulaciones, dando paso al tipo de ***MoCap Electromecánico*** o simplemente ***MoCap Mecánico.***

Otro tipo de captura de movimiento es la que se lleva a cabo por medio de sensores electromagnéticos, ***MoCap Electromagnético o MoCap Magnético***. Estos sensores pueden emitir una señal electromagnética con el estado de flexión de la articulación, esta señal es captada por un receptor electromagnético, para luego ser reproducida en forma de *ángulo articular*.

Mientras estas dos tecnologías iban en auge, el método de MoCap por visión también iba desarrollándose y volviéndose cada vez más efectivo.

Sistema de MoCap óptico.

Entre los tipos de MoCap es el más popular y el que mejores resultados ha arrojado a través de la historia. El constante avance tecnológico ha hecho de las cámaras instrumentos sumamente importantes ya que nos permite sacar imágenes cada vez más cerca de las tomadas por el ojo humano, en cuanto a su resolución, y sin duda mucho más rápidas que este grandioso órgano.

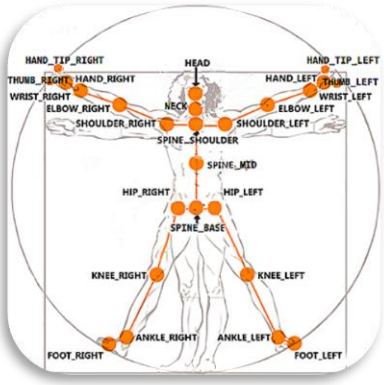


Figura 2 3 Puntos somatométricos del cuerpo que toma el Kinect. (Tomada de: [https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx)](https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx)

Podemos encontrar dos formas bien marcadas de captura de movimiento por medios ópticos, en primer lugar, encontramos el ***MoCap Óptico con Marcas:*** En el cual se disponen marcas por todo el cuerpo, en los puntos somatométricos (***Figura 2.3***), Estas marcas pueden ser a su vez de dos tipos, *Marca pasiva*, la cual está hecho de un material reflectivo que ilumina a un determinado ángulo de incidencia de la luz; estas marcas son tomadas por las cámaras que están dispuestas alrededor del área objetivo. Por el contrario, las *Marcas Activas* emiten su propia luz y de la misma forma las cámaras captan esta luz y las traducen en posiciones cartesianas a cada una de ellas, dando como resultado una *data* que relaciona cada posición articular. En la ***Figura 2.4*** se puede apreciar el funcionamiento de ambas. En la (a) vemos como las cámaras reflejan luz infrarroja para activar las marcas pasivas, en la (b) las marcas activas emitiendo luz.

a

b

Figura 2 4 a) Marcas pasivas emitiendo luz cuando esta choca con ellas. b) Marcas activas emitiendo luz por si mismas.

Otra forma de captura es la de ***MoCap Óptico Sin Marcas***, desarrollada a partir del surgimiento de los sensores de profundidad como el *Sensor Kinect de Microsoft*, sin ser este último el único, cuyo objetivo principal fue crear una herramienta que como resultado devolviera las características espaciales de los puntos somatométricos del cuerpo. La cámara de profundidad del *Kinect* permite tomar una data con las coordenadas cartesianas de cada punto que el patrón de luz emite. Un algoritmo de

tratamiento de datos o *nube de punto* reconoce el cuerpo humano y como resultados obtenemos la data llamada *"Esqueleto"* (Mirar **ANEXO 1).**

2.1.3. Estado del arte *captura de movimiento*

Captura de movimiento para reconocimiento de gestos

Easy gesture recognition for Kinect.
(Ibañez, Soria, Teyseyre, & Campo, 2014)**.**

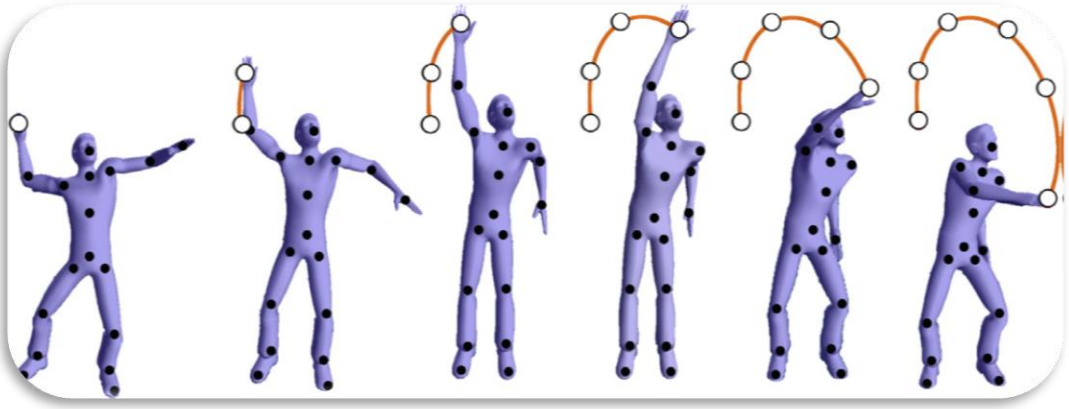


Figura 2 5 Secuencia de seis posiciones del sujeto y los puntos de su trayectoria. (Ibañez, Soria, Teyseyre, & Campo, 2014) (Jiang, Zhao, & Liu, 2014)

En esta aplicación vemos como se usan técnicas de aprendizaje automático para la detección de los gestos que se usan. Primero que todo se extraen el conjunto de puntos del Kinect, estos puntos se normalizan mediante el cálculo de su centroide por medio de la fórmula.

$$(\bar{X}, \bar{Y}, \bar{Z}) = \frac{\sum_{i=0}^{n}(X_i, Y_i, Z_i)}{n}$$

Luego se resta a los valores de cada punto este centroide, con el fin de hallar la correlación de los puntos de un conjunto de datos.

$$(X_i, Y_i, Z_i) = (X_i - \bar{X}, Y_i - \bar{Y}, Z_i - \bar{Z})$$

Todo esto con el fin de hacer invariante la trayectoria tomada, ya que los cambios que se producen pueden ser muy drásticos si el observador está en movimiento.

Posterior a esto se determina la técnica de aprendizaje que se va a usar y se comparan los resultados.

Accuracy of the Microsoft Kinect sensor for measuring movement in people with Parkinson's disease: (Galna, Barry, Jackson, Mhiripiri, & Oliver, 2014)

Este proyecto busca analizar el movimiento de personas con párkinson, de esta manera medirlo y estudiarlo más a fondo, para posteriormente encontrar una correlación con el proceso neural de la enfermedad. Este proyecto supone una buena alternativa de uso de estas técnicas de captura de movimiento.

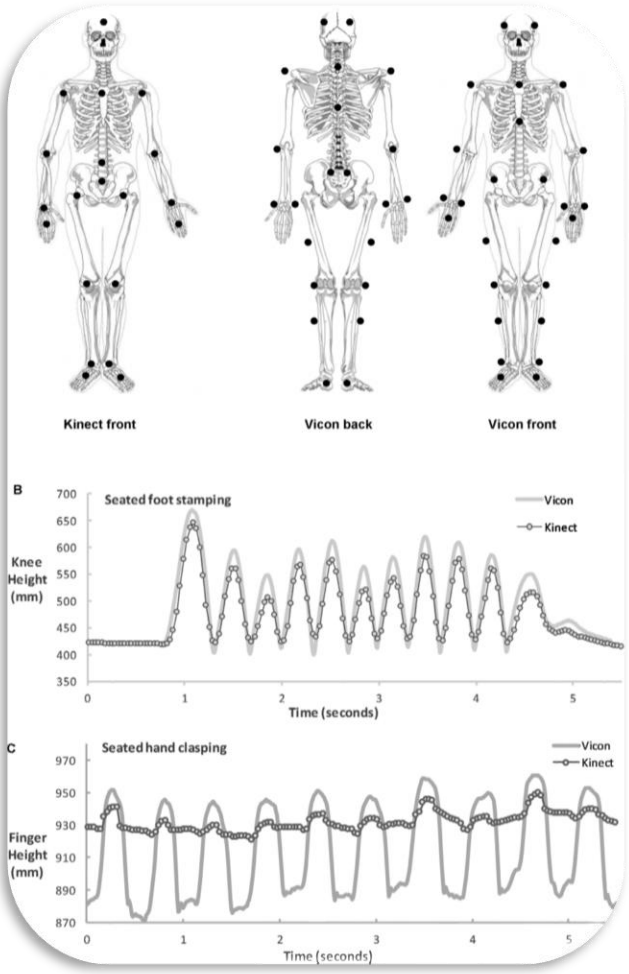


Figura 2 6 Imagen de muestra: (Galna, Barry, Jackson, Mhiripiri, & Oliver, 2014)

Validity of the Microsoft Kinect for assessment of postural control. (Clark, y otros, 2012).

Otro proyecto interesante que se puede revisar, en este se captura la postura del cuerpo y hace una corrección en ella. Es decir, que hasta que el paciente no esté ubicado de manera adecuada este no dará eventos positivos.

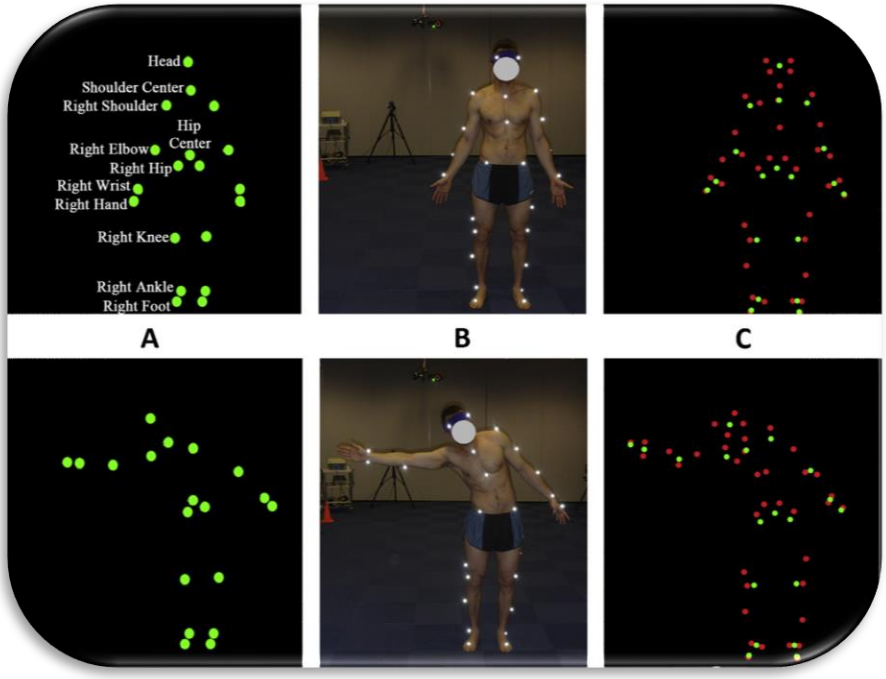


Figura 2 7 Imagen de muestra. (Clark, y otros, 2012)

2.2. PRESENTACIONES DE ALTO IMPACTO

2.2.1. ¿Qué es una presentación de alto impacto?

Según (Definición ABC, s.f.) , una presentación, se refiere al proceso a través del cual el contenido de un tema determinado se expondrá ante un auditorio u otro lugar. Su objetivo principal es el de transmitir una idea e informar a las personas acerca de un tema en cuestión.

2.2.2. Metodología de una presentación de alto impacto

Muchos emprendedores y blogueros coinciden en que para el desarrollo de una buena presentación son necesarios seguir ciertos parámetros entre los cuales, se incluyen planeación analógica, elaborar las preguntas correctas, elaboración de una historia, etc. Otros insisten en temas como la puntualidad, la presentación personal y otra clase de cosas, como la creación de diapositivas con la información pertinente de lo que se expone. (Navedo, 2009).

Por su parte, en el campo de las neurociencias se encuentra un campo llamado neuro-oratoria, la cual parte del análisis del funcionamiento del cerebro para lograr el éxito requerido en una presentación (Klarić, 2015). Klarić, define que la transmisión de la

información, es un sube y baja de tonos auditivos que fomentan en el que escucha interés o no por lo que se está diciendo. Este traza 10 puntos clave para la efectividad de una presentacion asi:

La entonación.
La entonación es la primera característica de la comunicación que conecta con un cerebro receptor.

El Cuerpo.
Significa el 50% de nuestra presentación. Debido a que es nuestro cuerpo el que logra expresar con mayor concordancia las ideas que queremos reproducir. De esta forma es necesario que no se ubique en el cuerpo ningún elemento que impida el movimiento completo de este.

La Ropa.
El hecho importante de la ropa no es en si el tipo de prenda que lleves, lo importante es el color que lleves, ya que lo que se busca con la ropa en una presentación es realzar los movimientos y las expresiones para la hora de comunicar.

Presentación
Esta debe ser lo más sencilla posible, y sin sobrecargo de información. Las metáforas son importantes. Es más fácil para el cerebro receptor establecer mayor atención por medio de metáforas.

El tiempo.
No hay un modelo para el spam de atención por temas. Sin embargo, hay que tener mucho cuidado con los tiempos.

Conserva tu estilo propio.
La autenticidad motiva mayor inspiración en los que atienden.
Tu Interacción con el público es necesaria.

Tu mensaje.
El mensaje es clave importante y debe ser conciso y entendible.

Tu credibilidad.

Para tener credibilidad debes tener buen manejo del tema, esto hace que lo que diga el orado se difunda con mejores resultados, Ya que una información mal manejada podría dejar dudas en las personas.

Tu.

Nos daremos cuenta de que las neurociencias tienen una visión bastante más amplia de lo que se involucra en una presentación.

2.2.3. Estado del arte de herramientas para presentaciones de alto impacto

Pasadores manuales.

a *b*

Figura 2 8 Pasadores manuales convencionales. (a) PowerPoint Remote Control Presentation Clicker es una herramienta hecha expresamente para controlar PowerPoint b) Genius Láser Media Pointer: Es una herramienta de la empresa Genius para el control general de diapositivas http://www.pceverest.com/productos/accesorios/apuntadores/genius-media-pointer-100-con-laser/

Estas herramientas son usadas comúnmente por las personas que ejercen esta actividad de manera cotidiana, es versátil y ofrece bastantes ventajas.

Capturadores de gestos para control de dispositivos.
MYO (MYO, s.f.)

Figura 2 9 A la izquierda vemos el dispositivo MYO en su presentación sin usuario, a la derecha vemos la disposición del dispositivo en el antebrazo de un usuario (Tomado de web oficial https://www.myo.com/)

Este dispositivo promete ser uno de los más completos, su nombre proviene del prefijo anglosajón para referirse al músculo, debido a que toma una muestra de las señales electromiográfícas del brazo y los traduce a eventos de ratón y eventos gestuales.

2.3. GRÁFICOS POR COMPUTADORA Y ANIMACIÓN 3D

2.3.1. Gráficos por computadora: Definición

Los gráficos por computadora se pueden definir como el trabajo de arte gráfico que se realiza de manera digital por computadoras. Estos son creados por medio de software y programas especializados que conjugan cálculos matemáticos y matriciales para representar en tres dimensiones un objeto determinado. Los gráficos por computadoras pueden ser en bidimensionales y tridimensionales, siendo los bidimensionales los primeros en desarrollarse, con la introducción de las computadoras análogas y las estaciones de trabajo especializadas más adelante se hablará de esto. Difieren entre sí por la forma como son generados, es decir, mientras un gráfico en 2D guarda un vector bidimensional de los puntos de una línea respecto a un sistema coordenado, un gráfico tridimensional es capaz de guardar en forma de vectores los puntos que componen un plano en el espacio. Cabe anotar que las gráficas en tres dimensiones son compuestas de planos en el espacio.

Estos han tenido furor en las bellas artes y han sido utilizados en la producción cinematográfica, sin embargo, ha sido en campos como la arquitectura e ingeniería, donde se han desarrollado las técnicas matemáticas más especializadas, en cuanto a simulación y análisis, pero sigue siendo en la industria cinematográfica donde se ha puesto a prueba su capacidad con la muestra de animaciones casi reales (Gutiérrez Lira, 2006).

2.3.2. Animación: Definición

La animación es la técnica que se utiliza para dar sensación de movimiento a imágenes y dibujos o, a otro tipo de objetos inanimados. Es en sí misma una ilusión óptica que va más allá de los típicos dibujos animados.

Los cuadros se pueden generar dibujando, pintando o fotografiando los minúsculos cambios que se hacen rápidamente por el evento en movimiento, a estos cuadros se les conoce como fotograma, estos guardan un evento en un tiempo determinado, los estándares para estos fotogramas se encuentran en la toma de 12, 24 y 30 por segundo, siendo los últimos el resultado de las técnicas actuales de fotografía.

2.3.3. Gráficos por computadora: Una breve historia

Los primeros experimentos con gráficos por computadoras fueron a fines de los años 40´s y principios de los 50´s, entre sus mayores precursores se destaca John Whitney. En un principio era solo un tema usado por ingenieros y con fines científicos e investigativos, esto a principios de los 60´s cuando las computadoras entraron en furor. Durante esta década también su uso se extendió al campo del arte.

Profundizando un poco más John Whitney fue un animador estadounidense, compositor e inventor, es bien conocido como el padre de la animación por computadora. Una antigua computadora análoga sirvió para que él y su hermano, James crearan una serie de películas experimentales a partir de un dispositivo con servomotores que controlaban el movimiento de las luces –sin duda fue el primer ejercicio de control de fotografías en movimiento-. Uno de los trabajos más reconocidos de Whitney en este período fue los títulos de la película Vértigo de Alfred Hitchcock en 1958. Ya en 1968 Whitney funda su empresa Motion Graphics Inc, que se encargaba de producir títulos para películas y televisión.

Las computadoras programables fueron una gran ayuda al desarrollo de imágenes digitales, una de las primeras fue la SEAC (The Standards Eastern Automatic Computer), diseñada por el pionero de la computación Russell Kirsch en 1950. Las técnicas de procesamiento de imagen se reducían a estudios de variación de intensidad de las imágenes y otras composiciones, simples en la actualidad, pero que en ese momento eran de lo más avanzadas. Uno de los ejemplos que se pueden dar de estas técnicas fue el escáner de tambor, el cual servía precisamente para medir las variaciones de intensidad en la superficie de una fotografía (Primera imagen digital). La fotografía mostraba al hijo de Kirsch en un recuadro de solo 176X176 píxeles, siendo esta la primera imagen escaneada. Junto a su equipo Kirsch utilizaban la computadora para extraer dibujos lineales, contar objetos, reconocer letras y proyectar

imágenes digitales en una pantalla de osciloscopio. Simplemente avances precursores de la imagen digital como tal. (Alemañ Baeza, 2014).

Entre los años 60´s y 70´s en la Universidad de Utah se crea la facultad de ciencias computacionales por David Evans, y muchas de las técnicas básicas de 3D fueron desarrolladas a principios de los 70´s con el proyecto ARPA (Advanced Research Projects Agency). Entre las técnicas que podemos encontrar están los Sombreados (Shading) Gouraud, phong y Blinn, mapas de texturas (Texture Map), algoritmos de determinación de cara oculta, Subdivisión de superficies (Sub Surface), Trazado de líneas en tiempo real y los primeros intentos de realidad virtual.

El final de los años 70´s y principios de los 80´s fueron prometedores para los gráficos por computador, puesto que las técnicas iban complementándose y el trabajo conjunto de los equipos de desarrollo en todo Estados Unidos se iba materializando con mejores resultados, se habría el camino para la graficación en 3D.

2.3.4. Gráficos en 3D

Es la técnica usada para la representación de objetos orgánicos e inorgánicos que se hace completamente por computadora. Estos consisten en el cálculo matemático avanzado que representa espacialmente un grupo de vértices, ejes y planos para la construcción de un objeto, estos planos o polígonos están tratados con una serie de algoritmos que determinan las propiedades de su cara visible (determinación de vector normal de visibilidad, reflexión y refracción de luz, sombreado, color y textura, etc). Una vez el objeto está conformado, el software también es capaz de modificar a gusto la malla.

Estos modificadores son técnicas matemáticas avanzadas de tratamiento y análisis de datos, que sirven para la conformación y formación de objetos, y posteriormente para su animación. De la misma forma que sirven para la adecuación y adhesión de texturas.

2.3.5. Rigging

Es la técnica usada para hacer *rigs*. Los rigs son sistemas de cadenas de huesos virtuales u otros objetos modificadores, con o sin características interactivas (restricciones), que sirven para definir deformaciones sobre objetos geométricos virtuales. Los *rigs* ofrecen un sin número de características y restricciones que pueden ser un trabajo bastante sencillo o simplemente uno complejo.

Al ser sistemas de cadenas pueden ser tomados como estructuras mecánicas que pueden calcular cinemáticas, dinámicas, entre otros tipos de comportamientos físicos. Estos modificadores son parte esencial a la hora de animar objetos virtuales. (Gutiérrez Lira, 2006)

2.3.6. Estado del arte animación de personajes/avatares 3D con Kinect.

Animaciones con Kinect.

Observation-oriented silhouette-aware fast full body tracking with Kinect (Jiang, Zhao, & Liu, 2014).

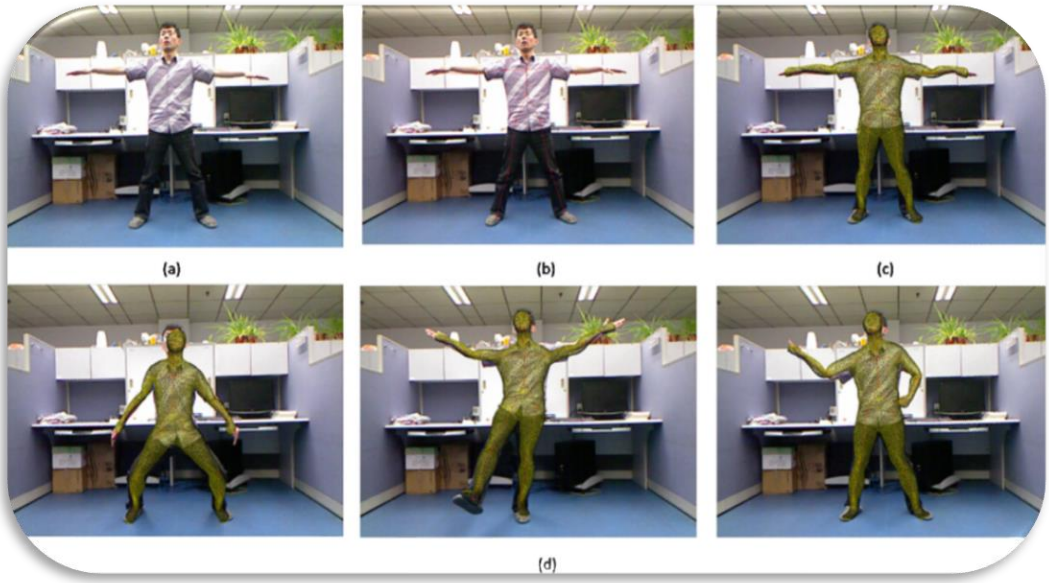


Figura 2 10 (a) User in Kinect RGB video stream. (b) Rectified skeleton using inverse kinematics (IK). (c) Tracking result for user shown in (a). (d) Three more tracking results under the movement of limbs. (Jiang, Zhao, & Liu, 2014)

Este no es precisamente un trabajo de animación con Kinect sin embargo coincide con algunas de sus características como lo es la utilización de objeto hecho en 3D, con el objetivo de superponerlo en la posición del usuario en la imagen RGB.

En esta técnica los desarrolladores, analizaron el contorno de la nube de puntos arrojada por el Kinect y la posición del esqueleto.

Turkish Sign Language Animation with Motion Capture (Söylev & Mendi, 2014)

Este proyecto realizado en el 2014 supone una comparación entre un sistema experto de captura de movimiento y uno no muy convencional aplicando Kinect, con el fin de establecer cual es más efectivo a la hora de hacer una animación. Dando como resultados datos no muy distintos.

Kinect driven 3d character animation using semantical skeleton (Wang, Ma, & Wang, 2012).

Este proyecto corresponde a una aplicación definitiva de un sensor Kinect para la animación de un personaje en 3D.

El sistema captura el movimiento y lo reproduce en un personaje en 3D.

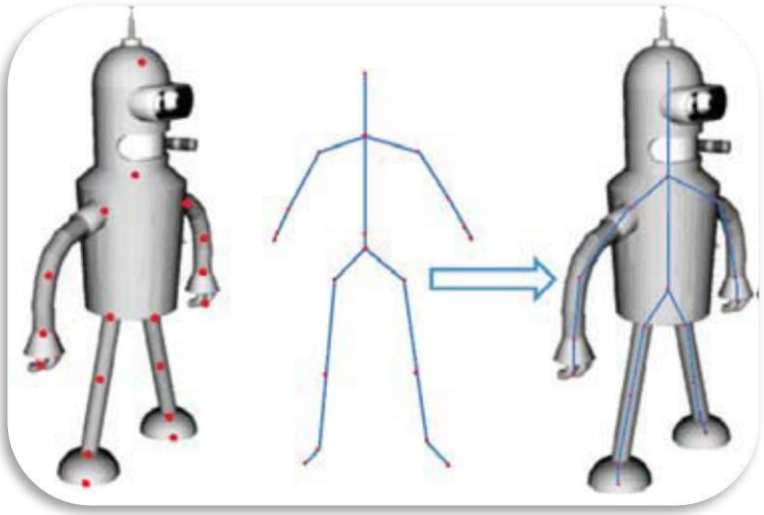


Figura 2 11 Embedding a semantics skeleton inside the 3D (Wang, Ma, & Wang, 2012)

CAPITULO 3.
ANÁLISIS, DISEÑO Y DESARROLLO DEL SISTEMA

3.1. ANÁLISIS.

Se necesita construir un sistema de captura de movimiento que sea capaz de proveer la relación de ángulos de las extremidades superiores e inferiores; además de hacer el seguimiento de los puntos de las muñecas para el reconocimiento de gestos.

Por un lado, se usan los ángulos para ser reproducidos en una estructura robótica virtual, con el fin de hacer una animación compleja de un elemento. Por lo cual se necesita realizar el modelado en 3D del personaje que se va a animar y adherirle una estructura modificadora llamada esqueleto. Al conjunto de personaje y esqueleto se le llamará *"robot virtual"*.

Con el seguimiento constante de las muñecas se construirán gestos que controlarán un grupo de acciones que se irán definiendo de acuerdo con las necesidades que se vayan presentando.

En ambos casos se necesita adquirir los conocimientos necesarios de las diferentes herramientas que se van a utilizar, por lo cual esta sección se dividirá en tres grupos las temáticas para identificar a grandes rasgos el uso de estas herramientas: *Captura de movimiento, Animaciones y Presentaciones.*

Se mira cómo usar debidamente la herramienta de programación PyKinect, para el uso y control de nuestro sensor Kinect y también se observan los aspectos generales de la animación 3D y los aspectos que hay que tener en cuenta para controlar las presentaciones.

Con esto se abarca la temática de una forma organizada para que a la hora del diseño se pueda proponer una solución adecuada para el sistema que se quiere obtener.

3.1.1. Captura de movimiento: Generalidades.

Hardware.

En cuanto a *hardware*, se utilizará un sensor KINECT el cual va a servir para escanear los distintos puntos somatométricos de un usuario dispuesto frente de este. Debido a las especificaciones del KINECT se debe definir un área de trabajo específica teniendo en cuenta el campo de acción del dispositivo. Un equipo de cómputo con el software necesario para el desarrollo de esta aplicación.

Sensor Kinect: Skeleton Tracking.

El *skeleton tracking* es simplemente la disposición de los puntos estequiométricos del cuerpo arrojada por el Kinect, estos son devueltos en formas de puntos con sus coordenadas de posición en 3D (***Figura 3.1***). El Kinect puede devolver la posición y el skeleton tracking de hasta 6 personas que estén en su rango de visión. También puede tomar solo la mitad superior de la información.

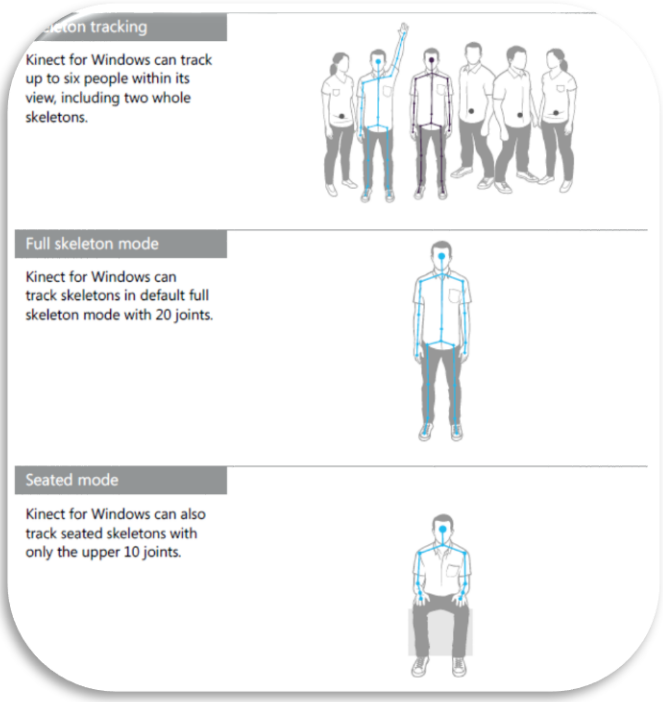


Figura 3 1 Muestra del Skeleton Tracking del Kinect. (Microsoft Corporations, 2013)

Rango de acción del sensor: Espacio de trabajo.

Según el **ANEXO 1**, el Kinect tiene un rango de acción descrito en la ***Figura 3.2*** se toma en cuenta el rango del ángulo de visión horizontal y vertical del KINECT para trazar el diseño del espacio de trabajo.

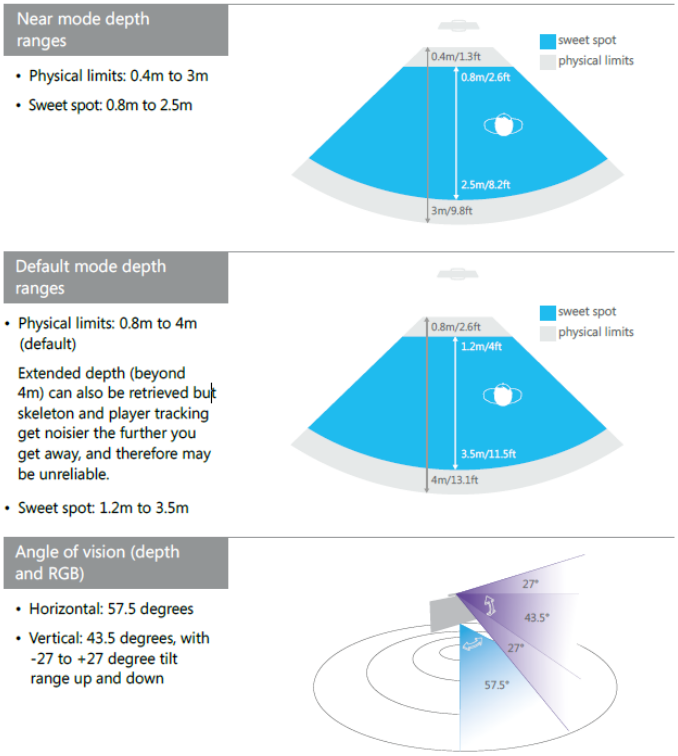
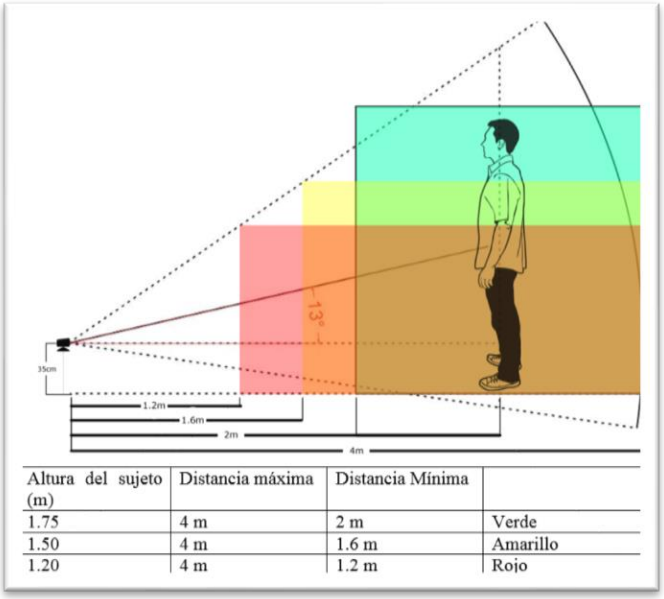


Figura 3 2 Rangos de acción del Kinect. (Microsoft Corporations, 2013).

El ángulo vertical de acción del KINECT será aproximado a 43° para hacer un cálculo prudente, y como se quiere que el dispositivo no sea tan visible a la vista del público entonces este se fijara a una altura de 35 cm con una inclinación positiva de 13° que será ajustada por el motor de inclinación del KINECT.

Como el rango de visión vertical del Kinect es de 43°, se tendrá que con una inclinación de 13° y a una distancia de 3 m la altura máxima alcanzada por el rango de visión vertical va a superar los 2 m a una distancia de 3 m. Así a medida que el individuo se acerque al KINECT este rango de visión va a ir disminuyendo, siendo 2 m una distancia apropiada para un límite mínimo de visión de un sujeto de aproximadamente 1.75 m de altura. Dejando otras posibilidades para sujetos con alturas más pequeñas, según lo esperado. Cabe anotar que estos cálculos se sacaron de forma gráfica, representando los distintos rangos de acción y los tamaños de los sujetos, como se muestra en la ***Figura 3.3***.



Altura del sujeto (m)	Distancia máxima	Distancia Mínima	
1.75	4 m	2 m	Verde
1.50	4 m	1.6 m	Amarillo
1.20	4 m	1.2 m	Rojo

Figura 3 3 Análisis respecto al espacio de trabajo, (autor)

Software.

El desarrollo de este sistema se hace en el lenguaje de programación de alto nivel Python, este es un lenguaje interpretado o de script que se ejecuta utilizando un programa intermedio llamado intérprete (Duque, 2010). Su ventaja se encuentra en su flexibilidad y su portabilidad; posee características similares a los lenguajes compilados. La flexibilidad de Python se debe a la disposición de una vasta cantidad de módulos que facilitan el desarrollo de un sin número de aplicaciones en cualquier campo de la programación de computadores.

El lenguaje Python se puede escribir y compilar desde múltiples intérpretes, siendo el intérprete en C, llamado CPython el más utilizado. PythonXY incluye por defecto este intérprete, así como otras herramientas (Módulos) que pueden ser necesarios en cuestiones específicas, en la red hay un sinnúmero de ejemplos que pueden guiar al programador a introducirse en este mundo. Para el desarrollo con Kinect es necesario la utilización de módulos de visualización y tratamiento de imágenes, así como un módulo que controle los eventos emanados del SDK propio del Kinect.

En la investigación se encontró que, Linux ofrece variedad de módulos de desarrollo para Kinect desde Python y múltiples SDK de distribución libre, sin ser de igual forma en Windows. Habiendo muy pocas opciones, se encontró una solución de desarrollo en Windows de Kinect con Python, pero involucra una serie de programas propios de Visual pero que son de uso libre para los programadores.

Control del sensor: PyKinect

Es un módulo para Python desarrollado por Microsoft de uso libre, el cual permite controlar y obtener los datos del Kinect desde Python. Dispone de dos módulos principales, uno que controla todos los eventos de las cámaras y del patrón de luz estructurada (Nui) y otro que controla los eventos de los micrófonos del Kinect (Audio). Su instalación se encuentra referida en el **ANEXO 1**.

Para iniciar el sensor Kinect hay que importar de la librería PyKinect el módulo "NUI" que contiene todas las opciones del sensor: Inicio, captura de cada frame para visualización de cámaras (nube de puntos) y control del motor.

```
from pykinect import nui
kinect = nui.Runtime()
```

Al hacer la asignación anterior hemos iniciado el Kinect y ya está listo para que lo comencemos a utilizar.

Visualización del sensor.

Para visualizar las cámaras del sensor Kinect desde Python es necesario construir una función que cargue la imagen RGB, la imagen IR y el Skeleton. Estas funciones van a ser llamadas en hilos independientes y van siempre a escribir sobre variables globales (*Skeletons*, *video* y *depth*). Para tal asunto, se describen en el programa principal las funciones llamadas *video_frame_ready*, *depth_frame_ready* y *skeleton_frame_ready*, quienes tienen la misión de llenar las variables antes mencionadas, se aclara que las funciones se asignan a los métodos con los mismos nombres que se encuentran dentro del Kinect, *Kinect.video_frame_ready*, *Kinect.depth_frame_ready* y *Kinect.skeleton_frame_ready* respectivamente, en la declaración del objeto kinect.

video_frame_ready.

Esta función recibe una variable global llamada *frame*, que contiene la información de las imágenes de las cámaras del Kinect.

```
def video_frame_ready (frame):
    global video
    video = np.empty( ( 480, 640, 4 ), np.uint8 )
    frame.image.copy_bits( video.ctypes.data )
    cv2.imshow('VIDEO', video)
```

Se asigna una matriz de 480x640x4 a la variable *video*, esta matriz contendrá valores nulos (0), ya que posteriormente va a ser llenada con los datos de la imagen RGB. Con el método *frame. image.copy* se le asignará a la matriz vacía (video.ctypes.data) los datos de la imagen RGB. Ahora solo queda mostrar la imagen y opcionalmente guardarla, esto se hace con los métodos de OpenCV, *cv2.imshow() y cv2.imwrite()* respectivamente. Se concluye mostrando una ventana con las características que se le dió con anterioridad al objeto *ventana* llamado 'VIDEO' que se construyó en nuestro main.

Esta función será asignada, posteriormente al método con el mismo nombre en el objeto Kinect.

```
Kinect.video_frame_ready += video_frame_ready
```

depth_frame_ready.

Al igual que en la función anterior, *depth_frame_ready* recibirá la variable global *frame*, debido a que, como se dijo anteriormente, esta contiene la información de las cámaras.

```
def depth_frame_ready( frame ):
    global depth
    depth = np.empty( ( 240, 320, 1 ), np.uint16 )
    rame.image.copy_bits( depth.ctypes.data )
    cv2.imshow( 'PROFUNDIDAD', video )
```

Esta función tiene prácticamente la misma lógica que la anterior, cambia en definitiva el tamaño de la ventana que se mostrará y el tipo de variable que se va a contener en depth.

Luego se asigna al método correspondiente en el objeto Kinect.

```
Kinect.depth_frame_ready += depth_frame_ready
```

Skeleton_frame_ready.

Esta también recibirá la variable *frame,* pero esta vez se le asignará simplemente a la variable global "skeletons" la data del Skeleton Tracking (SkeletonData), que se encuentra contenida en "frame".

```
def skeleton_frame_ready( frame ):
    global skeletons
    skeletons = frame.SkeletonData
```

Cabe comentar que la variable “skeletons” después de la asignación dentro de la función será un objeto que contendrá la información del número de personas que se encuentren dispuestos en frente del Kinect y las respectivas coordenadas de los puntos estequiométricos del cuerpo.

```
Kinect.depth_frame_ready += depth_frame_ready
```

Obtención del SkeletonTraking.

Obtención de los puntos estequiométricos del Skeleton Tracking.

Para acceder al Skeleton Tracking se ha dispuesto una clase llamada *Skeleto* esta clase contendrá todo al respecto del esqueleto. Entonces, para hacer el llamado del esqueleto se encuentra una variable del método “nui” al interior de PyKinect llamada *JointId*, esta contiene una lista de todos los puntos del cuerpo, en el ejemplo se muestra una lista con los puntos que corresponden a toda la espina, siendo el *Spine* el punto que dará con el ombligo del sujeto. Habiendo obtenido esta lista procedemos a hacer el respectivo llamado desde la clase Skeleto.

EJEMPLO

```
from pykinect.nui import JointId
SPINE = ( JointId.HipCenter,
          JointId.Spine,
          JointId.ShoulderCenter,
          JointId. Head )

class Skeleto():
          def __init__ (self, data, video):
          ################################
          ############ ESPINA ############
          ################################
          Espina   = data.SkeletonPositions[SPINE[1]]
          self.ES  =nui.SkeletonEngine.skeleton_to_depth_image( Espina,
                                                                640, 480 )
```

En el ejemplo anterior vemos que para llamar el vector coordenado que contiene la posición del punto de la espina será *data.SkeletonPositions*, al que se le suministra el nombre del punto que queremos obtener, en este caso SPINE[1] hace referencia a la segunda posición de la lista y que corresponde al objeto “Spine”. Después de haber hecho esto se hace una conversión del punto para poderlo graficar en una imagen de 2D. El convertidor *“nui.SkeletonEngine.skeleton_to_depth_image”* convierte las coordenadas cartesianas en posiciones en píxeles, con el fin de que puedan ser graficadas en una imagen 2D normal para que se retorne el valor para su posterior graficación.

Para llamar la clase “Skeleto” se tendrá que hacer antes una comprobación de que hay alguien dispuesto frente al Kinect, ya que si no se hace esto los contenedores de los puntos estarán vacíos y no se podrá hacer ningún tipo de cálculo matemático, marcando error a lo que el programa ingrese a las funciones anteriores. Para esto se escribe un comparador de la forma que se muestra a continuación.

```
if skeletons is not None:
        for index, data in enumerate(skeletons):
        if data.eTrackingState != nui.SkeletonTrackingState.TRACKED:
                continue
        PTo  = Skeleto(data, video2)
else:
        video2 = np.empty( ( 480, 640, 4 ), np.uint8 )
        cv2.imshow( 'VIDEO2', video2 )
```

La condición de este es que si “skeletons” no está vacío (If skeletons is not None) entonces proceda a extraer la data de “skeletons” por medio del bucle que se dispone para esto, dentro del “for” encontramos otro comparador cuya función es la de verificar el estado del Skeleton. Luego se procede a asignarle a una variable corta la clase “Skeleto” a la que se le introducirá la data extraída anteriormente y posterior a esto se procede a dibujar en la variable que contiene la imagen del frame actual. De esta forma se obtendrá el gráfico completo de todos los puntos del esqueleto en el espacio del Kinect.

3.1.2. Animaciones.

Calculo de los ángulos.

Se necesita crear un algoritmo que tome los puntos ergonómicos, somatométricos o articulares del cuerpo que devuelve el Kinect (SkeletonTraking), los vectorice y calcule la relación angular de las extremidades y la posición del “esqueleto”. Debe hacer un diccionario con esta información junto con el número de frame y el tiempo en que lo toma.

El programa tendrá que recibir el tiempo de duración de la película y grabara por ese tiempo los movimientos y tendrá que guardar dicho archivo para posteriormente ser reproducido en el robot virtual que se creará.

Además, deberá haber otro algoritmo en Blender 3D que reciba un archivo con la secuencia de movimiento tomada y los reproduzca en nuestro Robot virtual.

3.1.3. Presentaciones.

Gestos.

Según la RAE:

> *Gesto: "Movimiento de una parte del cuerpo, especialmente de la cara o de las manos, con el que se expresa algo."*

Se quiere controlar el paso de las diapositivas de una presentación con solo un movimiento gestual de una de las extremidades (o ambas), por esto es necesario construir un algoritmo que permita hacer el seguimiento de los puntos finales de las extremidades. Por el conocimiento previo que se tiene del Kinect este devuelve el punto de la palma de la mano y el de la muñeca y calcular si se produce un gesto o no, si se mueve las manos de manera intencional. Se debe identificar claramente la intencionalidad de un gesto y se tendrán que definir los movimientos que se quieren para el control de nuestras presentaciones.

Los gestos que se usen deben ser de alguna manera inusuales en la cotidianidad de los usuarios.

Paso de las diapositivas.

Para controlar las presentaciones se hace necesario un sistema que emule las acciones de teclado y mouse, conocimiento de la API de interfaz nativa de Windows (Win32api, win32con, win32gui).

Esta API de Windows ayudará a gestionar los archivos **.pptx* y las ventanas que se necesitan para nuestra aplicación. De la misma manera permite la gestión del teclado y mouse, para el desarrollo de la aplicación.

Otra opción para esto podría ser algunos módulos de la librería PyQt4, pero con algunas restricciones a la hora de conducir procesos a otras ventanas.

PyWin32.

Es una extensión de Windows para Python que permite controlar la Api de entorno de Windows, dando la posibilidad de hacer entornos de manera nativa en Windows, sin embargo, si se quiere utilizar en otro O.S no será posible ya que solo le pertenece a Windows. (ActiveState Docs, 2010)

PyQt4.

PyQt4 es una librería, adaptada para Python que permite desarrollar interfaces gráficas. Es una potente herramienta que contiene múltiples elementos utilizables.

Maneja base de datos, lista, tablas, gráficos, botones, checkBox, etc. (González Gutiérrez, 2008)

Los desarrollos hechos desde esta librería podrán ser usados en otros O.S´s y contarán con las características propias de las ventanas del sistema donde se encuentre.

3.2. DISEÑO.

3.2.1. Paradigma de programación.

En primera medida se usará el paradigma de programación estructurada, teniéndose en mira los Paradigma Orientado a Objetos (POO) y el Paradigma Orientado a Eventos (POE) para la solución de este problema, debido a que Python tiene dentro de sus características ser un lenguaje multi-paradigmas. También se tendrán en cuenta otras estrategias conforme vaya requiriendo este trabajo en el tiempo.

3.2.2. Descripción del sistema.

Aspectos generales de ambos sistemas.

Los sistemas capturan los datos frame a frame y traducen los puntos del *Skeleton tracking* en información eficiente para lo que queremos, es decir, toma los puntos somatométricos y calcula con ellos los ángulos para el movimiento de nuestro robot virtual (Descrito más adelante) y hace el seguimiento, recolección y diferenciación de los gestos que hagan con las manos.

Interfaz gráfica.

Se construyen dos interfaces gráficas con PyQt4 una para el control de presentaciones de alto impacto, la cual debe cargar una diapositiva previamente, visualizar el gesto hecho y controlar las pantallas de las ayudas visuales de nuestro sistema de presentaciones.

La otra es la interfaz gráfica de la aplicación para animaciones, este dispone de botones multimedia (Play, Stop, Pause y Rec), para controlar en que momento queremos grabar las acciones de nuestro cuerpo, tiene entre otras cosas un control por tiempo y la opción de mandar los datos en tiempo real hasta nuestro robot virtual. Puede generar un archivo *TXT* donde guarda las acciones grabadas, al tiempo que las visualiza en una tabla predeterminada.

Construcción de personajes.

Se necesita un personaje al que reproducirle los movimientos que capturamos con el Kinect, es por ello por lo que se modela y adecua un personaje para tal función. En este caso se tomará la imagen institucional de la Universidad de Pamplona y se hará lo debido para su construcción en 3D.

Para la construcción de personajes en 3D, es recomendable trazar primero un boceto de cómo lo queremos. Del boceto que se haga se sacarán al menos dos vistas oblicuas entre sí, en este caso la frontal y la lateral izquierda, para posteriormente proporcionárselas al programa en el que vamos a hacer el modelado. Nuestros personajes serán alzados en el entorno de trabajo de Blender 3D.

Boceto.

Como boceto se toma la imagen institucional de la Universidad de Pamplona. Esta ya contiene un amplio recurso para su trabajo.

Figura 3 4 Imagen corporativa de la Universidad de Pamplona. (www.unipamplona.edu.co)

En la ***Figura 3.4*** encontramos alguna de las imágenes proporcionadas en la página oficial de la institución.

Toma de imágenes.

Una vez bocetado lo que queremos, se forman varias imágenes del personaje y se correlacionan entre sí por medio de líneas que demarcaran las medidas más importantes y sobresalientes del personaje, por ejemplo: los ojos, nariz y boca. ***Figura 3.5.***

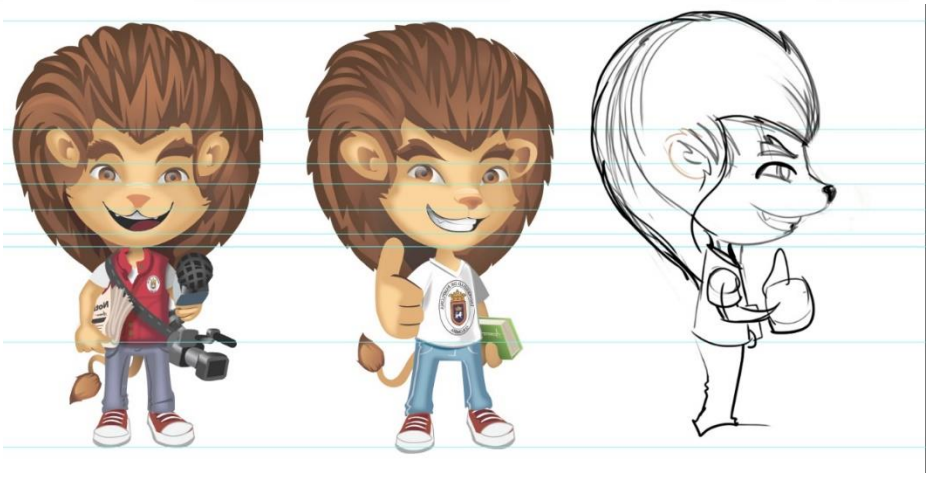


Figura 3 5 Imagen corporativa de la Universidad de pamplona. (www.unipamplona.edu.co y Alban Blanco)

3.3. DESARROLLO.

El desarrollo del sistema no goza de una linealidad determinada.

El paradigma de programación que se está utilizando es la Programación Orientado a Objetos (POO), esto hace que nuestro razonamiento sea dirigido hacia la resolución de este paradigma.

La información que veremos a continuación corresponde a esta lógica de la POO y no a la programación estructurada. También durante la construcción del proyecto se puede observar cómo se implementó el paradigma de programación orientado a eventos (POE).

3.3.1. Aspectos generales.

En condiciones iniciales (Ángulo de inclinación = 0°, altura de la base = 5 cm) el Kinect tiene un sistema de coordenadas de la forma como se muestra en la ***Figura 3.6,*** descrito por la matriz A_k:

$$A_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & .05 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

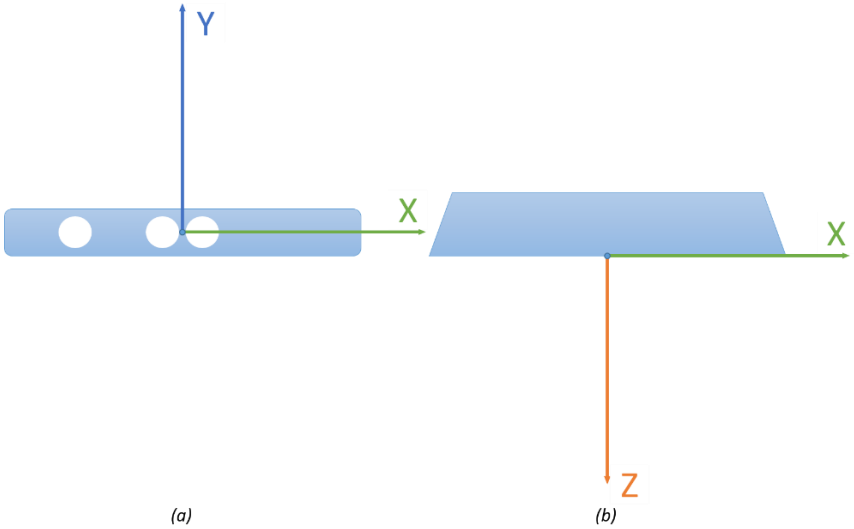


Figura 3 6 Sistemas coordenados del Kinect en condiciones iniciales: Altura de la base propia = 7 cm (a) vista frontal, (b) vista superior. (Autor)

Considerando que el Kinect tiene como variable el ángulo de inclinación *AngKinect*, y que estará afectado por la altura *AltKinect*, el sistema de la ***Figura 3.6*** se transforma en:

$$S_0 = A_k * Trasy(AltKinect) * Rotx\ (AngKinect) \qquad \textbf{(1)}$$

Para definir posteriormente un elemento llamado *'Eskeleto'*, producto del skeletonTraking' del Kinect, se debe representar los distintos puntos articulares del cuerpo en una transformación matricial homogénea que relacione los puntos entre sí para que tome la representación del esqueleto, precisamente este se definirá como el objeto *Eskeleto*. Este da una mejor comprensión y facilitará el análisis de los cálculos posteriores.

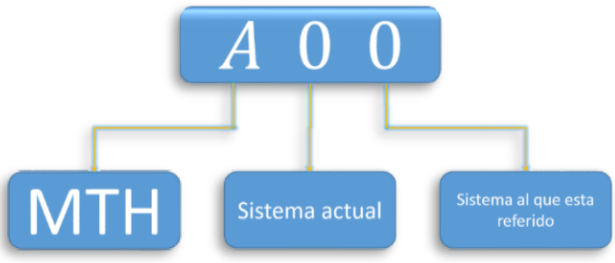


Figura 3 7 Convención usada en la programación para entendimiento de la matemática implícita. (Autor).

Los cálculos que se van a tomar en cuanto del objeto *Eskeleto* son los correspondientes a las distintas extremidades del cuerpo, la posición y la orientación del sujeto en el plano superior.

3.3.2. Animaciones.

Cálculos del "eskeleto".

Inicio: Construir el nuevo sistema coordenado

Se asigna la MTH representando el sistema S_0, en este caso del sensor *Kinect*, como se muestra en la ecuación **(1)**.

Posteriormente es trasladada hacia el punto de la Espina y se rota en contra del ángulo de elevación del Kinect, que este siempre esté con el eje "y" hacia arriba:

$$A_{Es}^{k} = Tras(P_{Es}^{K}) * A_k * \text{Rotx}(-\text{AngKin}) \qquad \textbf{(2)}$$

Una vez hecho esto se procede a calcular el vector en 2D que hay de hombro izquierdo a hombro derecho ($\vec{V} = \overrightarrow{V_{xz}}$) en el plano *X-Z;* de este vector se calcula la rotación para asignársela a la rotación respecto a *"y"* que toma el vector unitario *"x"* del sistema coordenado de la Espina. Como se muestra en la ***Figura 3.8***.

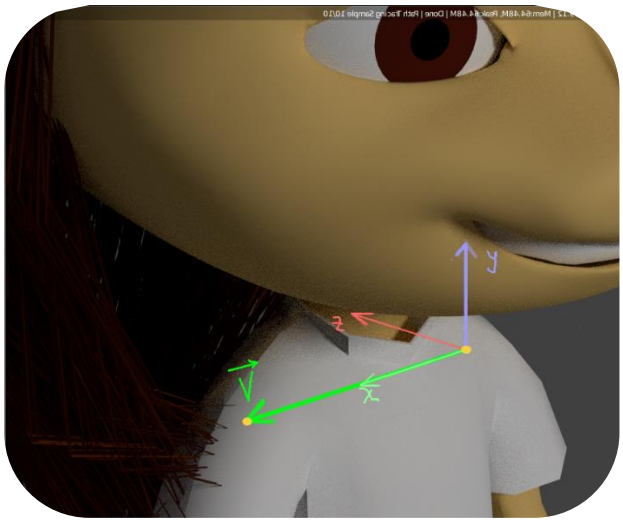


Figura 3 8 Visualización del vector en 2D que hay de hombro a hombro. (Autor)

Con la componente *"y"* de cada vector en cero se crea $\overrightarrow{V_{xz}}$. P

$$\overrightarrow{V_{xz}} = P_{Hd}^{Es} - P_{Hi}^{Es} \quad (3)$$

$$\theta = arcTan2\,(\overrightarrow{V_z}, \overrightarrow{V_x}) \quad (4)$$

Entonces nuestra nueva A_{Es}^{k} quedaría así:

$$A_{Es}^{k} = A_{Es}^{k} * Roty(\theta) \quad (5)$$

Una vez calculado la MTH de la Espina, se procede a referenciar cada punto hacia ella, ya que será el punto director del esqueleto.

$$A_{k}^{Es} = \left(A_{Es}^{k}\right)^{-1}$$

$$P_{n}^{Es} = A_{k}^{Es} * P_{n}^{k} \quad (6)$$

*Donde **n** es el índice del punto que se quiere referenciar.* Con esto se da por terminado el inicio del esqueleto.

Ángulos.

Una vez se halla referenciado cada punto en el esqueleto se procede a calcular los ángulos que se necesitan para el movimiento del robot virtual. Para esto se agrupan los puntos de forma secuencial y se definen 5 secciones en las cuales están las dos extremidades superiores, las dos inferiores y la sección central que va desde el centro de la cadera hasta el punto de la cabeza llamada espina. Los puntos donde se van a tomar los ángulos serían los correspondientes a los hombros, codo y muñecas en la sección de las extremidades superiores y, cadera, rodilla y tobillo en la sección de las extremidades inferiores, teniendo como inicio sus respectivos centros. Para efectos de abreviar y luego poder construir una función general en programa, que calcule los ángulos de los distintos componentes de las secciones se usan las convenciones ***Centro, HC, CR y MT*** de la siguiente manera:

Centro: *HombroCentro* o *CaderaCentro*
HC: *Hombro* o *Cadera*
CR: *Codo* o *Rodilla*
MT: *Muñeca* o *Tobillo*

Además, se podrían calcular algunos otros como el del grupo de la espina.
Para obtener los ángulos de cada componente de las extremidades se crea una función a la que, en términos generales, se le ingresan los puntos que componen las distintas secciones y que realiza el siguiente calculo:

1. Se vectoriza cada sección de las extremidades a partir de los 4 puntos que se ingresan, se crean Clavícula, Brazo y Antebrazo.

$$\overrightarrow{VecA} = HC - Centro\ (Clavicula)$$
$$\overrightarrow{VecB} = CR - HC\ (Brazo)$$
$$\overrightarrow{VecC} = MT - CR\ (Antebrazo)$$

2. Una vez vectorizados se asignan a los sistemas coordenados para hacer el análisis de las distintas rotaciones. Como se muestra en la ***figura 3.9****.*

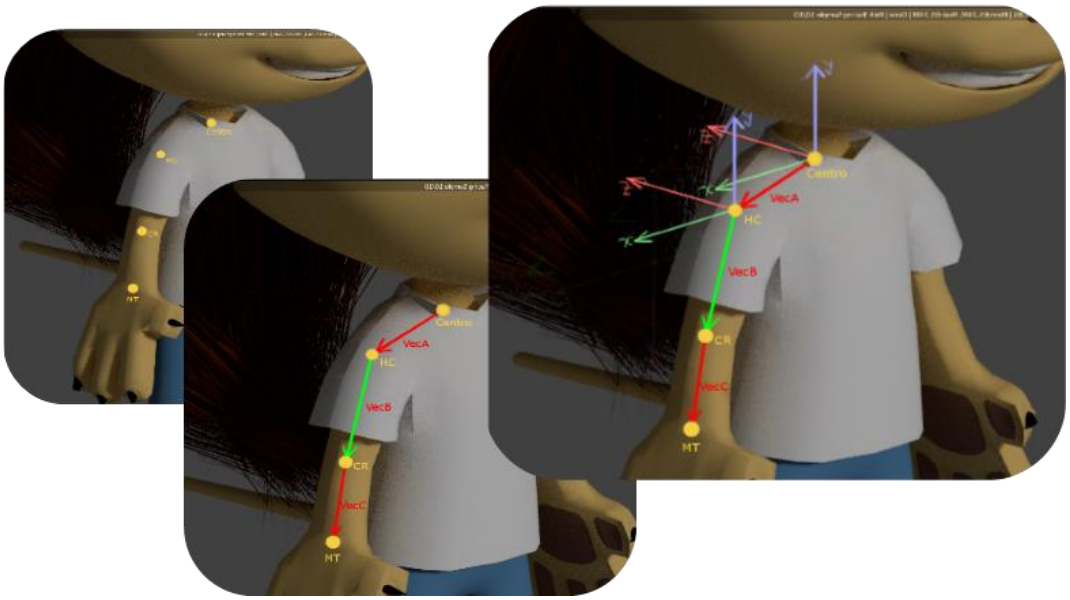


Figura 3 9 Visualización del cálculo del Skeleto. (Autor)

3. Se calculan los ángulos $TetaA, Teta1B, Teta2B$ *(****figura 3.10****)*

$$\boldsymbol{TetaA = arcTan2(-VecA_y, VecA_x)}$$
$$\boldsymbol{Teta1B = arcTan2(-VecB_y, VecB_x)}$$
$$\boldsymbol{Teta2B = arcTan2(-VecB_z, rB)}$$

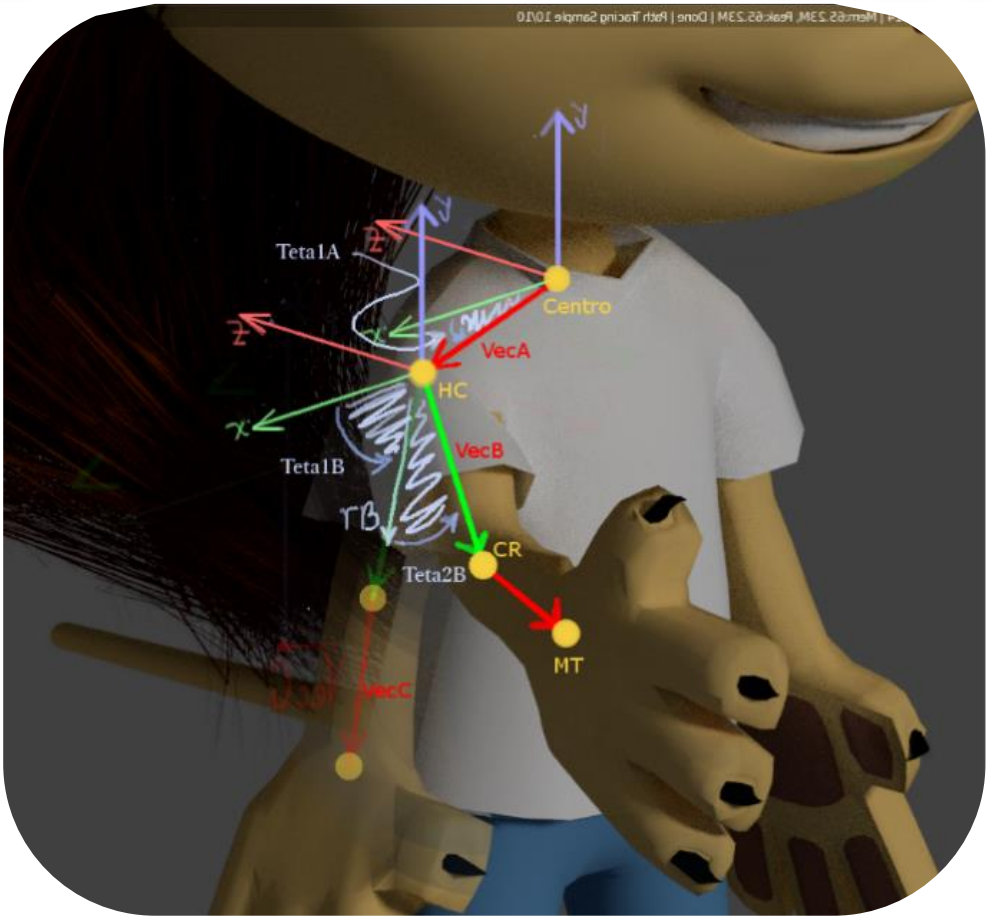


Figura 3 10 Cálculo de los ángulos de rotación del hombro. (Autor).

4. Se calcula la rotación del brazo en su propio eje, que sería sobre el mismo VecB, para ello crea un plano HC-CR-NT y hace el siguiente cálculo:

Se calculan los vectores unitarios de VecB y VecC, para posteriormente hallar $TetaC$.

$$U_{\overrightarrow{VecB}} = \frac{\overrightarrow{VecB}}{|\overrightarrow{VecB}|}$$

$$U_{\overrightarrow{VecC}} = \frac{\overrightarrow{VecC}}{|\overrightarrow{VecC}|}$$

$$Cos\ (TetaC) = U_{\overrightarrow{VecB}}\ dot\ \overrightarrow{VecC}$$

$$Sen\ (TetaC) = \sqrt{1 - Cos\ (TetaC)^2}$$

$$\boldsymbol{\theta C = arcTan2(Sen\ (TetaC), Cos\ (Teta\theta C))}$$

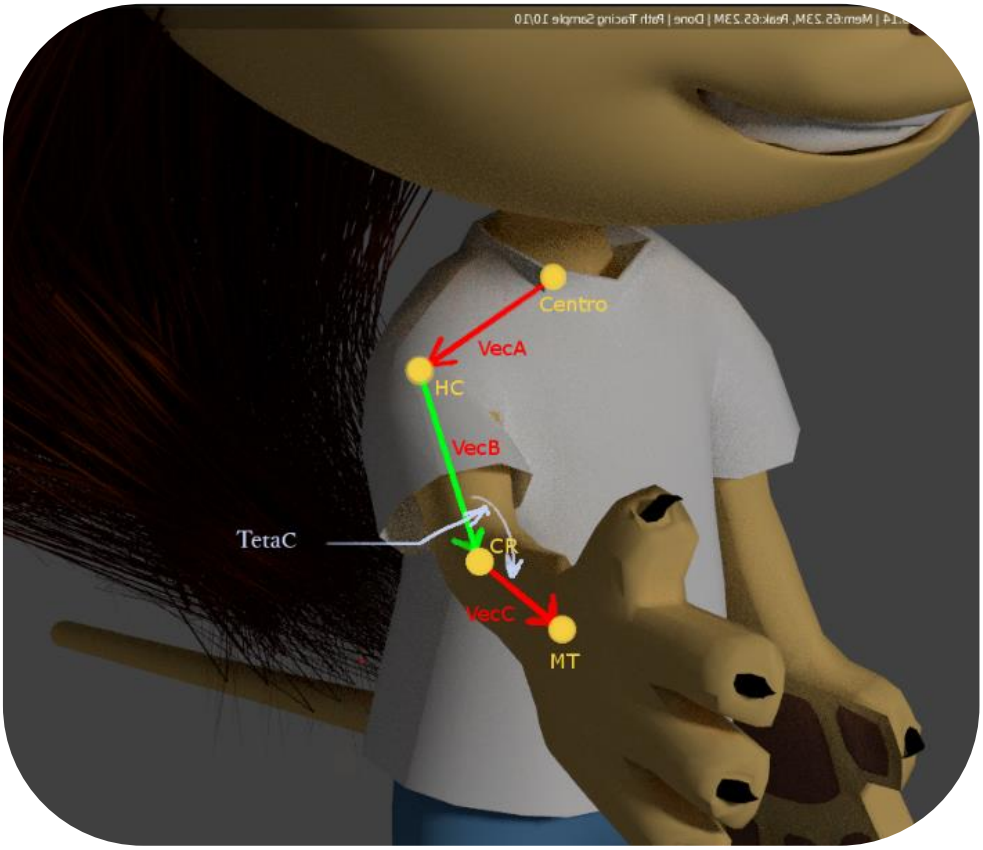


Figura 3 11 Cálculo del ángulo de rotación del codo. (Autor)

Se arman las MTH del hombro y el codo:

$$\boldsymbol{A_{HC}^{k} = A_{Es}^{k} * Tras(P_{HC}^{Es})}$$

$$\boldsymbol{A_{CR}^{k} = A_{Es}^{k} * Tras\left(P_{CR}^{Es}\right) * Rot([0, Teta2B, Teta1B])}$$

Quedando el sistema como en la ***figura 3.12***

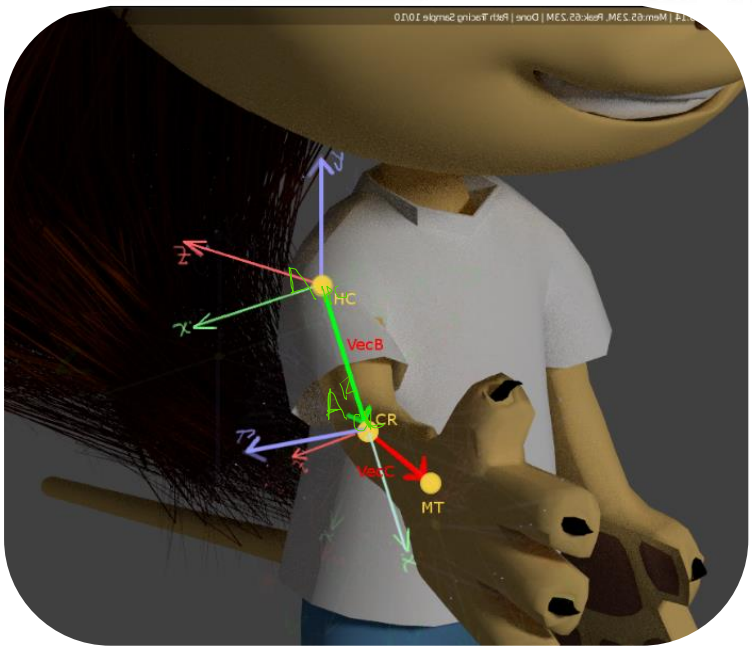


Figura 3 12 Ubicación del sistema coordenado del codo. (Autor).

5. Se calcula el ángulo que equivale a la rotación que tiene el plano HC-CR-NT alrededor de VecB, por lo cual se procede con el siguiente cálculo:

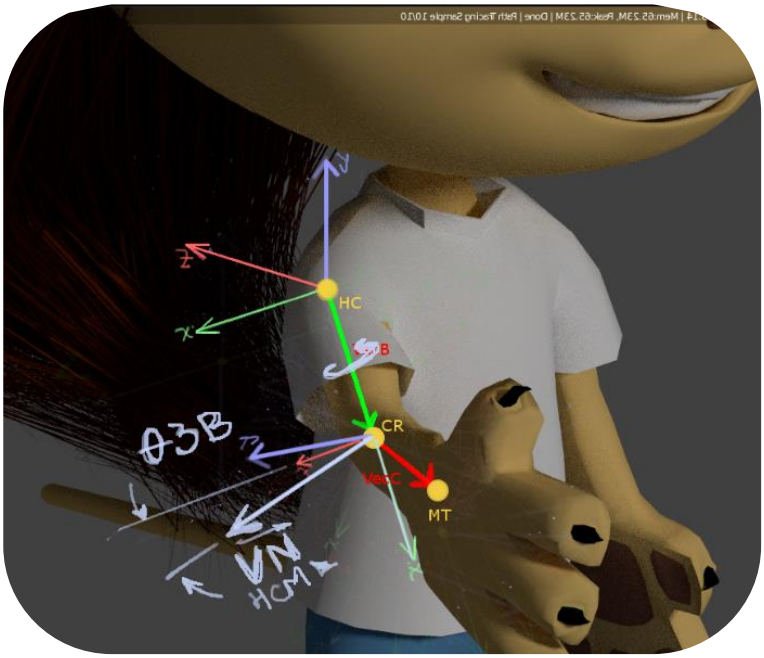


Figura 3 13 Cálculo para el ángulo de giro del brazo. (Autor)

Se calcula el vector normal al plano HC-CR-NT, $\overrightarrow{VN}$ Y lo convertimos en unitario $U_{\overrightarrow{VN}}$:

$$\overrightarrow{VN} = U_{\overrightarrow{VecB}} \times U_{\overrightarrow{VecC}}$$

$$U_{\overrightarrow{VN}} = \frac{\overrightarrow{\boldsymbol{VN}}}{\left|\overrightarrow{\boldsymbol{VN}}\right|}$$

$$Cos\,(Teta3B) = U_{\overrightarrow{VN}} \bullet X_{\boldsymbol{A^k_{CR}}}$$

$$\therefore\ X_{\boldsymbol{A^k_{CR}}} : Vector\ X\ de\ la\ Rot\ de\ \boldsymbol{A^k_{CR}}$$

$$Sen\,(Teta3B) = \sqrt{1 - Cos\,(Teta3B)}$$

$$\boldsymbol{Teta3B = arcTan2(Sen\,(Teta3B), Cos\,(Teta3B))}$$

6. Por último se construye la tupla de ángulos de la extremidad:

$$\boldsymbol{(TetaA, Teta1B, Teta2B, Teta3B, TetaC)}$$

Esquema de programación.

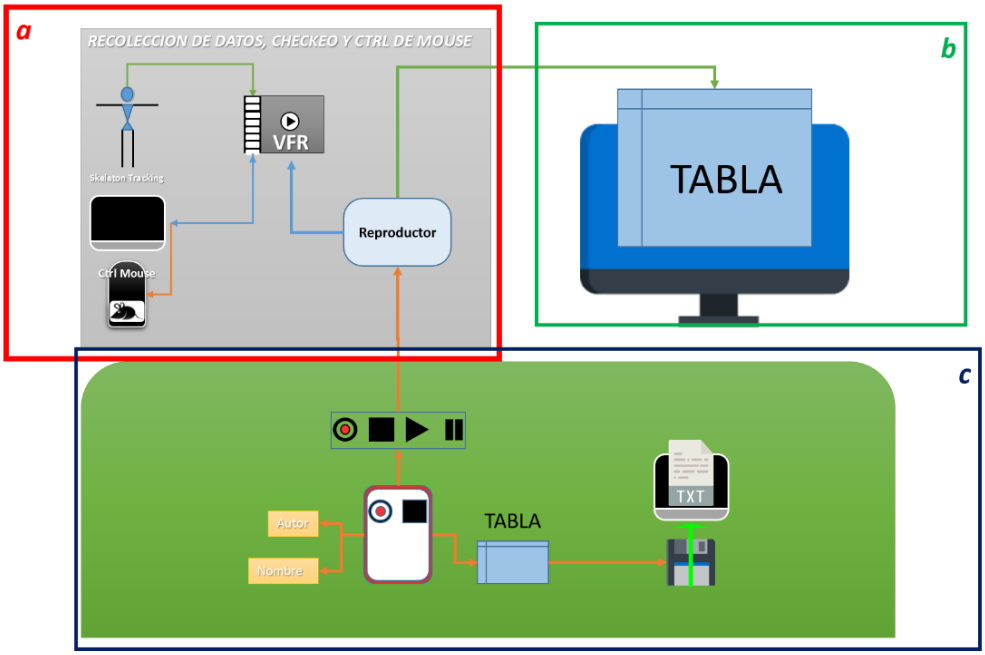


Figura 3 14 Diagrama completo de la estructura del programa. (a) Parte matemática, (b) Visualización e (c) Interfaz gráfica. (Autor)

Para efectos de poder entender mejor las distintas partes del programa se divide la estructura en varias: Parte Matemática, Visualización e Interfaz gráfica (Ver ***Figura 3.14)***.

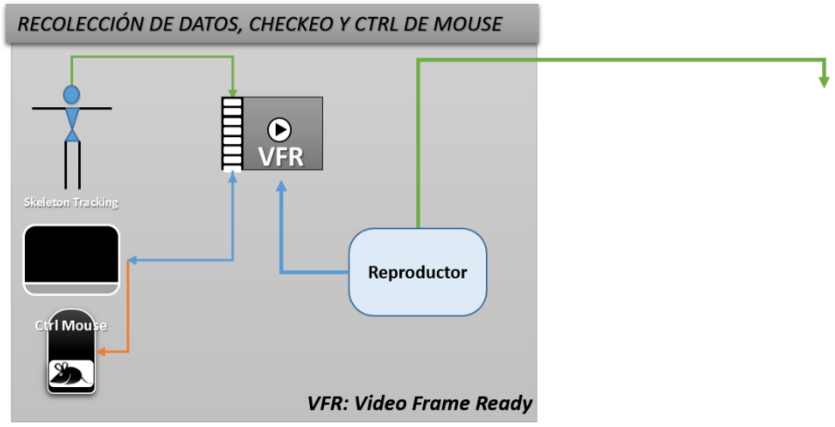


Figura 3 15 Representación de la parte matemática. (Parte (a) del diagrama general) (Autor)

Primero la parte de la matemática usada en el proyecto. Esta parte del programa hace la recolección, identificación, procesamiento y reproducción de nuestros datos

obtenidos ***Figura 3.15***; esta parte es la que se encarga de extraer todos los datos que necesitamos.

VFR, es el módulo donde se encuentran la función que se le asignará al *NuiVideoFrameReady* de la librería PyKinect. (Ver **ANEXO 6).** Esta última recibe los datos del esqueleto, Control mouse y las órdenes del reproductor.

Figura 3 16 Representación en el diagrama de la visualización de la tabla. (Parte (b) del diagrama general) (Autor)

Posterior a esto seguirá la parte de visualización. En esta parte se muestran los datos que se vayan recolectando, todo esto relacionado en una tabla que contiene: Frame, tiempo y los ángulos obtenidos en ese frame en concreto. (Ver ***Figura 3.16***).

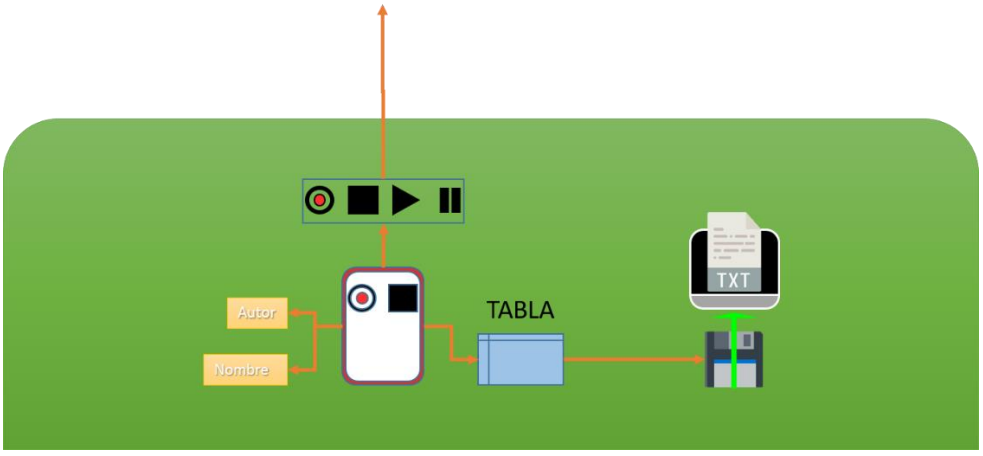


Figura 3 17 Esquema de la interfaz gráfica. (Parte (c) del diagrama general). (Autor)

La simplificación de lo anterior se verá representada en la interfaz la cual utiliza una función que hace la comunicación entre las distintas partes de nuestro programa ***Figura 3.14 (c)***, dicha función es la que controla todos los eventos, emulando para

ellos un control I/O. En esta función se toman las distintas órdenes para la gestión de nuestro programa (Ver ***Figura 3.17***).

3.3.3. Presentaciones.

Una vez analizado el problema, en cuanto a una herramienta que controle la presentación de diapositiva de manera remota por medio de gestos corporales, se procede a hacer los cálculos.

Cálculos del "Eskeleto".

El esqueleto en este caso tendrá el mismo inicio que en el anterior, junto como la parte de comprobación. Se adiciona una parte de seguimiento.

Comprobación.

La comprobación se hace de la siguiente manera:

1. Se determina si el sujeto se encuentra a la distancia recomendada para el uso del sistema (3 mt), se hace observando la coordenada en Z del punto de la Espina, esto me da la distancia lineal en metros que hay desde el sensor hasta el esqueleto, como se muestra en la ***Figura 3.18****.*

$$2 < \boldsymbol{Espina.z} < 4$$

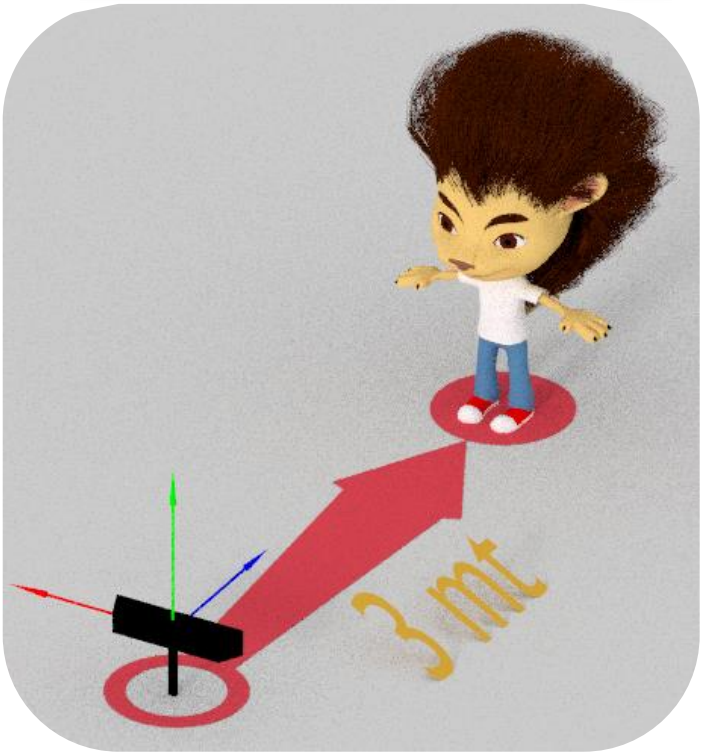


Figura 3 18 Disposición de la persona frente al Kinect para comprobación a una distancia de 3 metros. (Autor)

2. Una vez se cumpla con la condición anterior, es necesario extender los brazos hasta que el ángulo ***teta*** *sea menor de 45° ($\theta < 45°$), además la distancia que hay entre el Centro y las muñecas ($\overrightarrow{VD}$ y $\overrightarrow{VI}$,* ***Figura 3.19****) no podrá ser mayor que la suma de todos los componentes vectoriales que componen las distintas partes de la extremidad superior. Así:*

Se calculan los vectores $\overrightarrow{VD}$ y $\overrightarrow{VI}$:

$$\overrightarrow{VD} = MuneDer - Centro$$
$$\overrightarrow{VI} = MuneIzq - Centro$$

Se calculan sus módulos:

$$|\overrightarrow{VD}| = \sqrt{X^2_{\overrightarrow{VD}} + Y^2_{\overrightarrow{VD}} + Z^2_{\overrightarrow{VD}}}$$
$$|\overrightarrow{VI}| = \sqrt{X^2_{\overrightarrow{VI}} + Y^2_{\overrightarrow{VI}} + Z^2_{\overrightarrow{VI}}}$$

Hago la suma de los módulos de todos los componentes vectoriales del brazo.

$$Clavicula = Hombro - Centro$$
$$Brazo = Codo - Hombro$$
$$Antebrazo = Mune - Codo$$
$$\boldsymbol{MD = |ClaviculaDer| + |BrazoDer| + |AntebrazoDer|}$$
$$\boldsymbol{MI = |ClaviculaIzq| + |BrazoIzq| + |AntebrazoIzq|}$$

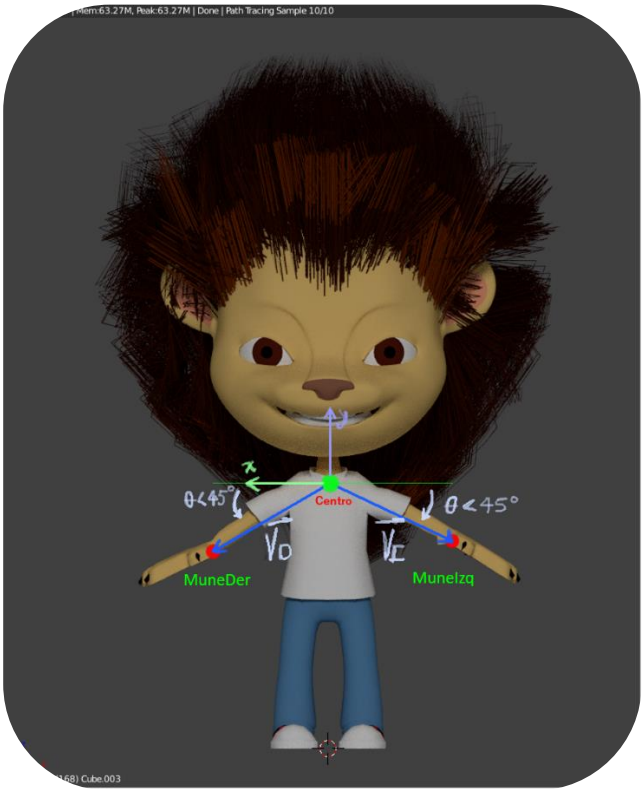


Figura 3 19 Cálculo de los ángulos para la comprobación. (Autor)

Así entonces el valor de $\boldsymbol{|\overrightarrow{VD}|}$ *no excederá el valor de* $\boldsymbol{MD}$*, ni será menor a este menos 10 cm.*

$$\boldsymbol{MD > |\overrightarrow{VD}| > MD - 10cm}$$

Los ángulos θ´s son:

$$\theta d = arcTan2(-Y_{\overrightarrow{VD}}, X_{\overrightarrow{VD}})$$
$$\theta i = arcTan2(-Y_{\overrightarrow{VI}}, -X_{\overrightarrow{VI}})$$

Y deberán ser menores a 45°:

$$\boldsymbol{\theta d|\theta i < 45°}$$

(Mirar la Figura 3.20).

Static_X.

Este es un elemento imaginario que se activa después de que está comprobado el esqueleto, hace que se active el recolector, es decir es un sensor de posición, así cuando la muñeca llegue a esa posición determinada se activa una bandera

Se hace una lista con todos los elementos comprobatorios, donde el limX y limY serán un porcentaje de la suma de las longitudes de los elementos de la extremidad superior, comprobadas en el paso anterior.

$$Lista = \left\{ \begin{array}{c} 'stc\_D\_X': MuneDer.x >= limX, \\ 'stc\_D\_Y': MuneDer.y >= 0 \,\&\, MuneDer.y <= limY, \\ 'stc\_D\_Z': MuneDer.z <= 0 \,\&\, MuneDer.z >= -limZ - 0.3, \\ 'stc\_Ad\_x': MuneDer.x >= limAX \,\&\, MuneDer.x <= limAX + 0.2, \\ 'stc\_Ad\_y': MuneDer.y >= limAY \,\&\, MuneDer.y <= limAY + 0.2, \\ 'stc\_Ad\_z': self.MuneDer.z <= -limAZ, \\ 'stc\_I\_X': MuneIzq.x \leq -limX, \\ 'stc\_I\_Y': MuneIzq.y >= 0 \,\&\, MuneIzq.y <= limY, \\ 'stc\_I\_Z': MuneIzq.z <= 0 \,\&\, MuneIzq.z >= -limZ, \\ 'stc\_Ai\_x': MuneIzq.x <= -limAX \,\&\, MuneIzq.x >= -limAX - 0.2, \\ 'stc\_Ai\_y': MuneIzq.y >= limAY \,\&\, MuneIzq.y <= limAY + 0.2 \\ 'stc\_Ai\_z': self.MuneIzq.z <= -limAZ, \end{array} \right\}$$

Figura 3 20 Disposición de los Static's en torno al cuerpo. (Autor)

*Después se definen un conjunto de reglas, de manera que cuando alguna de las muñecas pasa por alguno de los Static´s (**Figura 3.20**) esta eleva una bandera para que inicie la recolección del vector ventana y el vector de tiempos.*

Recolector.

En el Recolector se define el Vector Ventana. Este elemento recolecta un número determinado de posiciones de la muñeca desde el momento en que se activa un Static hasta pasados 1 segundo, recolectando las posiciones de por lo menos 26 puntos, equivalente a los 26 fps que debe captar el sistema. Es decir, que creará un vector de más o menos esa misma longitud que contiene los puntos de la muñeca en cada frame.

El recolector va ser guiado por un bucle que este comprobando si se ha producido un Static, y dentro contendrá la siguiente fórmula.

Llenado (P, Tant,Tact, VecVen, VecTim)

Donde:

***P**: Nuevo punto por asignársele al VecVen.*
***Tant** : tiempo inicial en el que se comenzó el conteo.*
***Tact:** Tiempo de la lectura actual.*
***VecVen:** Vector Donde se guardan los puntos.*
***VecTim:** Vector que contiene el tiempo de duración Entre frames.*

Al interior hace el siguiente cálculo:

$$t_i = (Tac - Tant) * 0.001\ sg$$
$$VecVen[i] = P_i$$

$$\boldsymbol{VecTim}[\boldsymbol{i}] = \boldsymbol{t_i}$$

Retornando como resultado el VecVen, VecTim y tiempo.

La forma de VecVen y VecTim es:

$$\boldsymbol{VecVen} = [(\boldsymbol{P_{0X}} \quad \boldsymbol{P_{0Y}} \quad \boldsymbol{P_{0Z}}), \quad (\boldsymbol{P_{1X}} \quad \boldsymbol{P_{1Y}} \quad \boldsymbol{P_{1Z}}), \quad (\boldsymbol{P_{2X}} \quad \boldsymbol{P_{2Y}} \quad \boldsymbol{P_{2Z}}) \cdots (\boldsymbol{P_{iX}} \quad \boldsymbol{P_{iY}} \quad \boldsymbol{P_{iZ}})]$$

Donde $\boldsymbol{i} \leftarrow$ *long(VecVen – 1)*
Junto a esto se tomará en cada frame el punto de la espina, este también será recolectado en un vector que contendrá el mismo número de posiciones que los vectores anteriores.

Cálculos para gestos.

Los gestos se producen por la comparación del absoluto de los diferenciales entre el punto inicial y el final en cada eje, es decir, que en el eje que tenga un margen diferencial mayor que en los otros, entonces sobre ese eje será el movimiento. La dirección (por ejemplo: Izquierda/derecha) se la dará la evaluación entre el punto inicial y el final, dicha distancia debe superar un mínimo estipulado. ***Figura 3.21***.

Figura 3 21 Imagen del capturado de puntos para los gestos. Donde $\boldsymbol{P_n}$ *es el punto mínimo. (Autor)*

Normalización de los puntos de la Espina.

En este paso, la totalidad de los puntos de la Espina, incluidos en el vector ventana, se normaliza, según lo descrito en (Ibañez, Soria, Teyseyre, & Campo, 2014), con las siguientes fórmulas:

$$(\bar{X}, \bar{Y}, \bar{Z}) = \frac{\sum_{i=0}^{n}(X_i, Y_i, Z_i)}{n}$$

$$(X_i, Y_i, Z_i) = (X_i - \bar{X}, Y_i - \bar{Y}, Z_i - \bar{Z})$$

El fin de esta acción es encontrar el centroide entre todos los puntos de la espina, será necesario referir a este centroide los puntos restantes del*eskeleto* de cada frame, así se tomarán datos más precisos para describir el gesto que se desea.

Identificador

El identificador esta siempre censando el estado del VecVen, de manera que cuando este ya se ha llenado por completo entonces procede a identificar el gesto. Incluso antes, para que el gesto tenga un tiempo para mostrarse.

En la ***Figura 3.21*** se ve como es este proceso. El Vector Ventana esta explícito por medio de los puntos que siguen la muñeca, después de entrar al *Static* comienza a guardar dichos datos. Al terminar pasa a calcular los diferenciales.

Diferenciales

Se calculan los diferenciales así:

$$P0 = VecVen[i]$$

$$P1 = VecVen[i-1]$$

$$\Delta X = |P1_x - P0_x|$$
$$\Delta Y = |P1_y - P0_y|$$
$$\Delta Z = |P1_z - P0_z|$$

Luego se compara cual es mayor, al tiempo que se revisa si supera el valor mínimo establecido, después se establece la dirección. Por ende, el Gesto.

Gesto: Derecha.

Se produce si $\Delta Y < \Delta X > \Delta Z$, y el punto inicial tiene que ser menor que el punto final en el VecVen. Mandara *True* si estas condiciones se cumplen.

Gesto: Izquierda.

Se produce si $\Delta Y < \Delta X > \Delta Z$, y el punto inicial tiene que ser mayor que el final en el VecVen. Mandara *True* si estas condiciones se cumplen.

Gesto: Arriba.

Se produce si $\Delta X < \Delta Y > \Delta Z$, y el punto inicial tiene que ser menor que el punto final en el VecVen. Mandara *True* si estas condiciones se cumplen.

Gesto: Abajo.

Se produce si $\Delta X < \Delta Y > \Delta Z$, y el punto inicial tiene que ser mayor que el punto final en el VecVen. Mandara *True* si estas condiciones se cumplen.

Control de mouse

Esta parte del algoritmo para control de presentaciones hace un seguimiento de la muñeca y cuando llega a una zona determinada controla el cursor del computador. Es posible dar *Click*, *DobleClick*, entre otras funciones que se pueden realizar con el mouse del ordenador.

Esquema de programación.

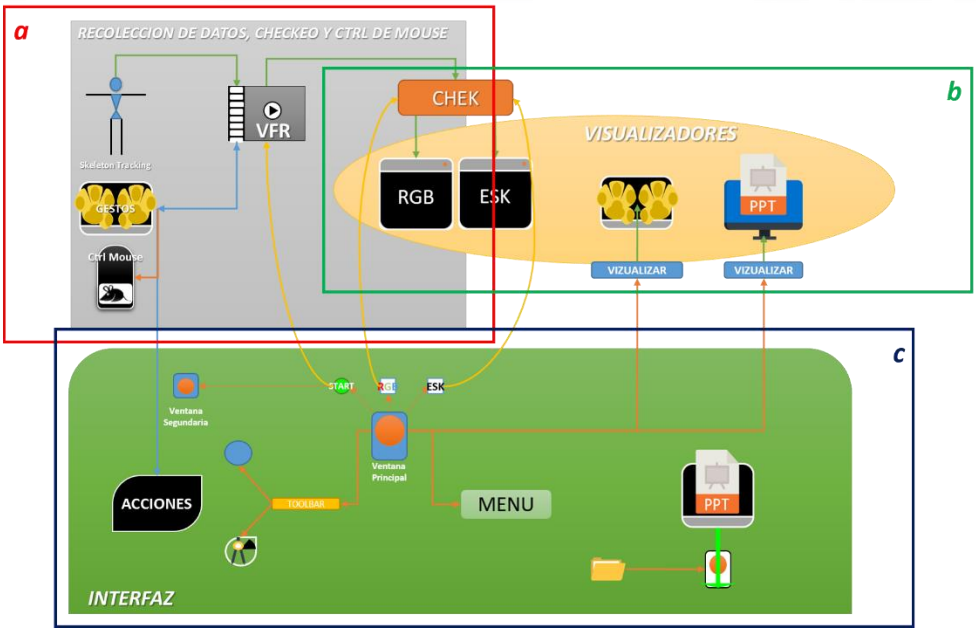


Figura 3 22 Esquema general de la programación. (a) Matemática, (b) Visualizador, (c) Interfaz gráfica y lógica computacional. (Autor)

La programación va en varios aspectos. Por una parte, está toda la matemática implícita, esta parte de nuestro programa hace la recolección, identificación y chequeo de los gestos, como se muestra en la ***Figura 3.22 (a)***; Esta parte es la que se encarga de extraer todos los datos que se necesitan, se puede notar con mayor detalle en la ***Figura 3.23***.

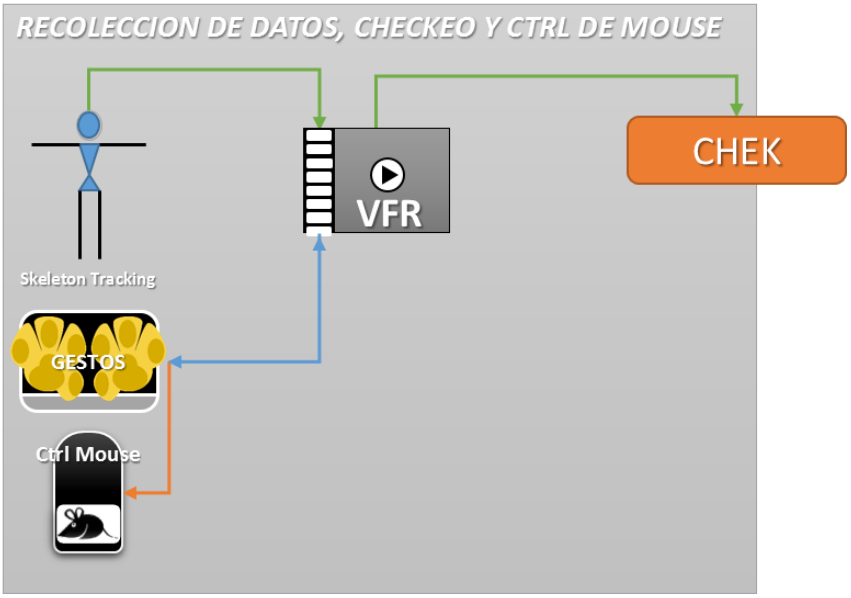


Figura 3 23 Esquema de la sección de recolección, identificación y chequeo. (Autor)

En segundo lugar, está la parte de visualización, en esta parte del programa se va a visualizar las imágenes que arroja el Kinect, las diapositivas y los gestos. Aunque es una sección que deriva en conjunto de la recolección y de la interfaz, que se verá posteriormente, se pone por aparte ya que es necesario describirla individualmente. Esta parte visualiza todos los datos ya recolectados y procesados. ***Figura 3.24.***

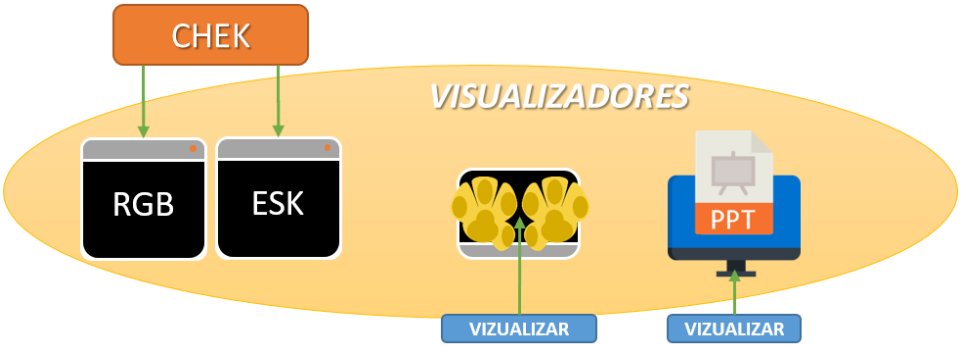


Figura 3 24 Esquema para la visualización de los distintos elementos del programa.

Por último, la parte de la interfaz, en esta parte (Y no de manera explícita) se encuentra una función que hace la comunicación entre las distintas secuencias, esta función es la que controla los eventos entre programas, emulando un control I/O. En los anexos se encontrará evidencia de esto. También aquí es donde se producen las acciones de cada gesto que se quiera usar. Ver ***Figura 3.25***.

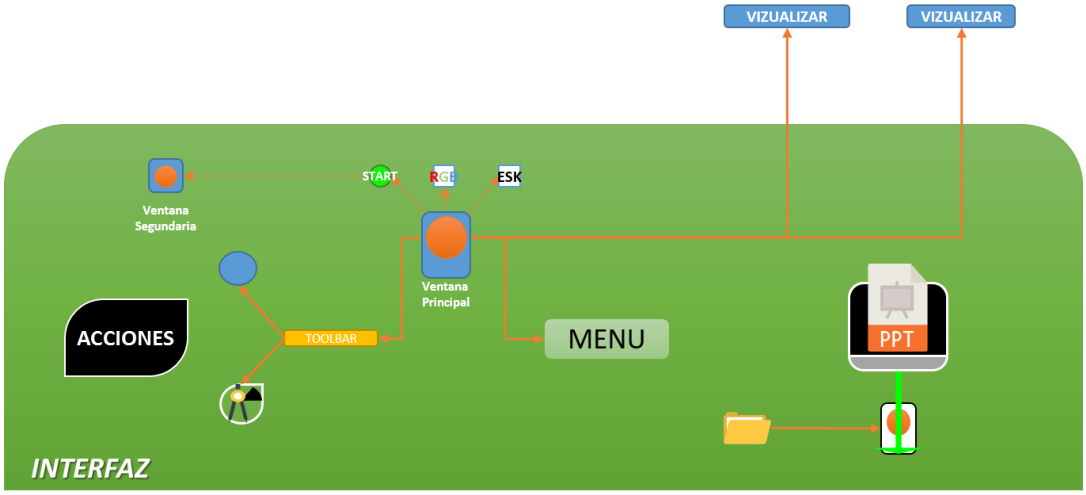


Figura 3 25 Esquema de la interfaz gráfica. Autor

Metodología para presentaciones de alto impacto.

La metodología es la descrita en el ANEXO 3.

3.3.4. Construcción de personajes.

Los personajes se realizarán en el entorno de trabajo de Blender 3D. ANEXO 4.

Modelado.

En primer lugar, se realizó un análisis del modelo a usar y se traza la imagen en por lo menos dos planos el frontal y el lateral. Si se quiere utilizar información más completa se puede hacer, esto involucra por lo general nubes de puntos.

Modelado del rostro.

Una vez la imagen esta lista, esta se dispone en el *BackGround* del programa en sus diferentes vistas, siendo de mayor importancia en este caso las vistas *frontal* y *lateral derecha*, las imágenes deben estar dispuestas en el centro según su vista, es decir *Lateral derecha de la imagen* en el centro de la *lateral derecha del espacio de trabajo de Blender*, así con las *Vistas frontales*, como se muestra en la ***Figura 3.26***.

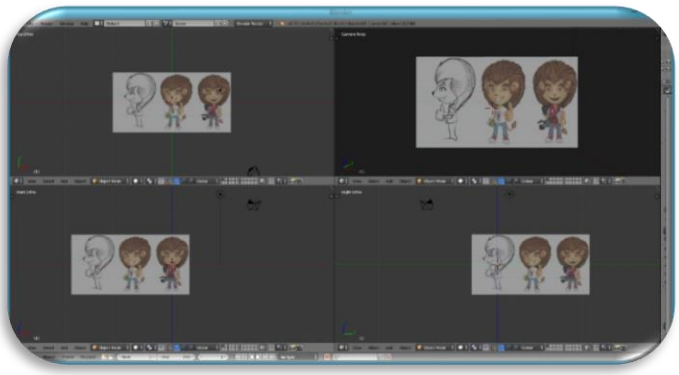


Figura 3 26 Espacio de trabajo en Blender 3D con las distintas imágenes en el BackGround de las diferentes vistas. (Autor).

Una vez dispuesta como se muestra en la Figura 3.27 anterior, se procede primero modelando la cara del personaje. Para ello usaremos una curva del tipo *Bézier*, y se comienza a modelar las curvas más significativas del rostro, como lo son: *contornos frontal y lateral de la cara (**Figura 3.27-a y Figura 3.27-b**), ojos y orejas (**Figura 3.27-c**).*

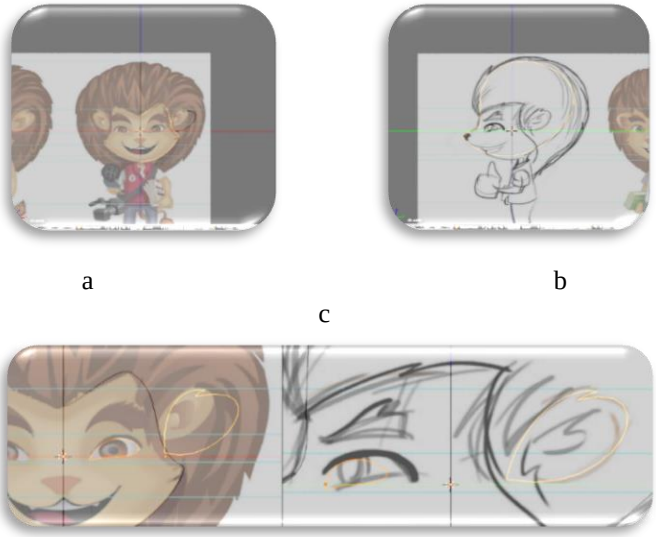


Figura 3 27 Imágenes Background en vistas a) Frontal, b) lateral y c) acercamiento de ambas vistas contrastadas. (Autor).

Posteriormente se procede a construir la malla, que es la representación de la superficie de todo el rostro.

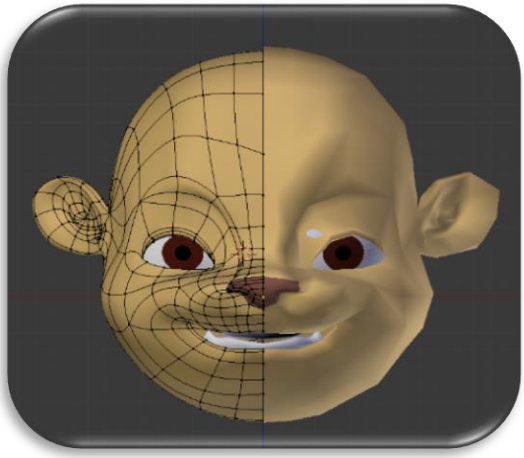


Figura 3 28 Rostro terminado, contraste entre una superficie rustica y el modificador SubSurf. (Autor).

En la ***Figura 3.28*** se muestra, a la derecha el resultado de las superficies sin ningún tipo de tratamiento, a la izquierda, se muestra ya el resultado con el *"Mirror"* y el *"Subsurf"*. Que son dos tratamientos útiles a la hora de modelar objetos con simetría igual y el otro nos permite hacer la subdivisión de cada plano para que el objeto en

general tome una forma mucho más suave. Otros tratamientos que se pueden aplicar influyen en como refleja la luz la superficie y le dan armonía a la composición.

Modelado del cuerpo.

Un proceso parecido, al anterior, se utiliza para modelar el cuerpo del personaje y con algunas técnicas de suavizado de superficies se obtendrán mejores resultados más adelante. ***Figura 3.29.***

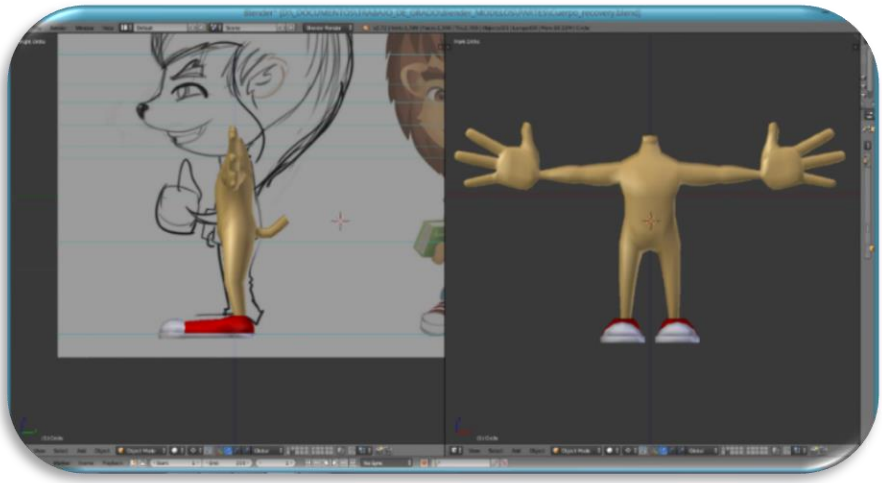


Figura 3 29 Modelo del cuerpo del personaje. (Autor)

Modelado de los ojos y los dientes.

Para este caso los ojos y la boca se modelarán aparte. ***Figura 3.30***.

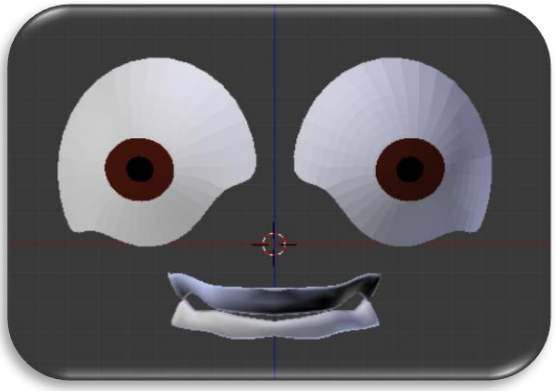


Figura 3 30 Modelo de los ojos y los dientes del personaje. (Autor)

Posible resultado.

Uno de los posibles resultados que se pueden obtener se encuentra en la ***Figura 3.31***, se puede notar como también fue modelado el cabello y la ropa.

Figura 3 31 Personaje con la ropa modelada sobre él. (Autor).

Construcción del rigging "Eskeleto".

El *Eskeleto* es una estructura virtual que actúa como modificador de los vértices de una malla determinada según el rango de acción que este tenga o el grupo de vértices asignados a cada hueso (Como ha de llamarse cada tramo de la estructura). Este supone la estructura antropomórfica del modelo, para su animación. (***Figura 3.32***.).

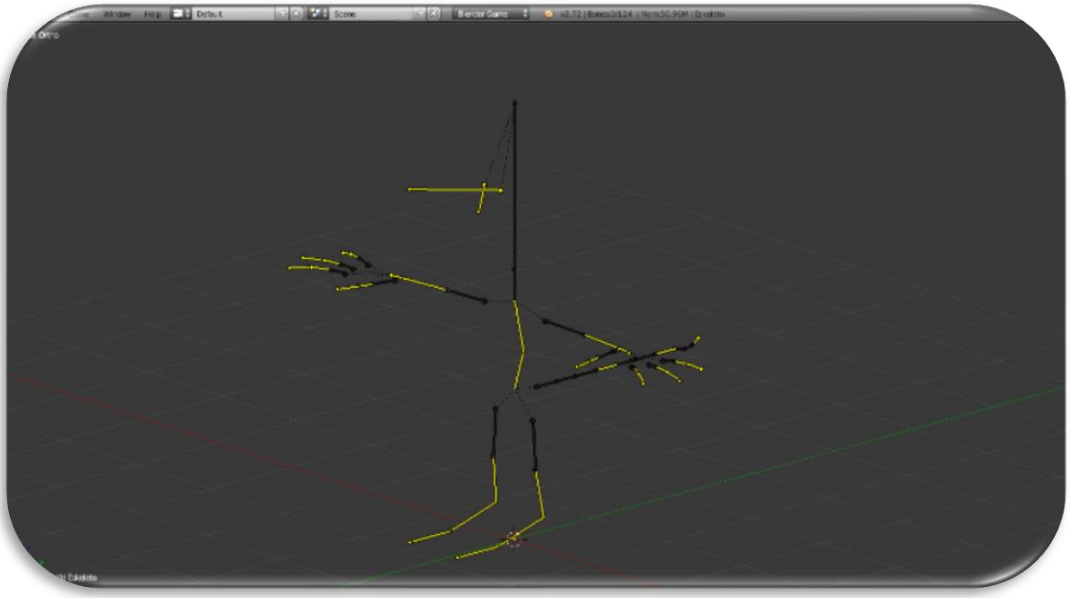


Figura 3 32 Esquema general del objeto Eskeleto. (Autor)

Detalles del rigging "Eskeleto".

El *"Eskeleto"* está compuesto por una serie de estructuras que representan los brazos, piernas, cabeza (Con ojos y boca) y tronco. A este va asignado todo un sistema de control (Otros huesos debidamente relacionados), que controlan las extremidades, cabeza y tronco, así mismo como la cola del personaje.

Extremidades.

Rigging de los Brazos:

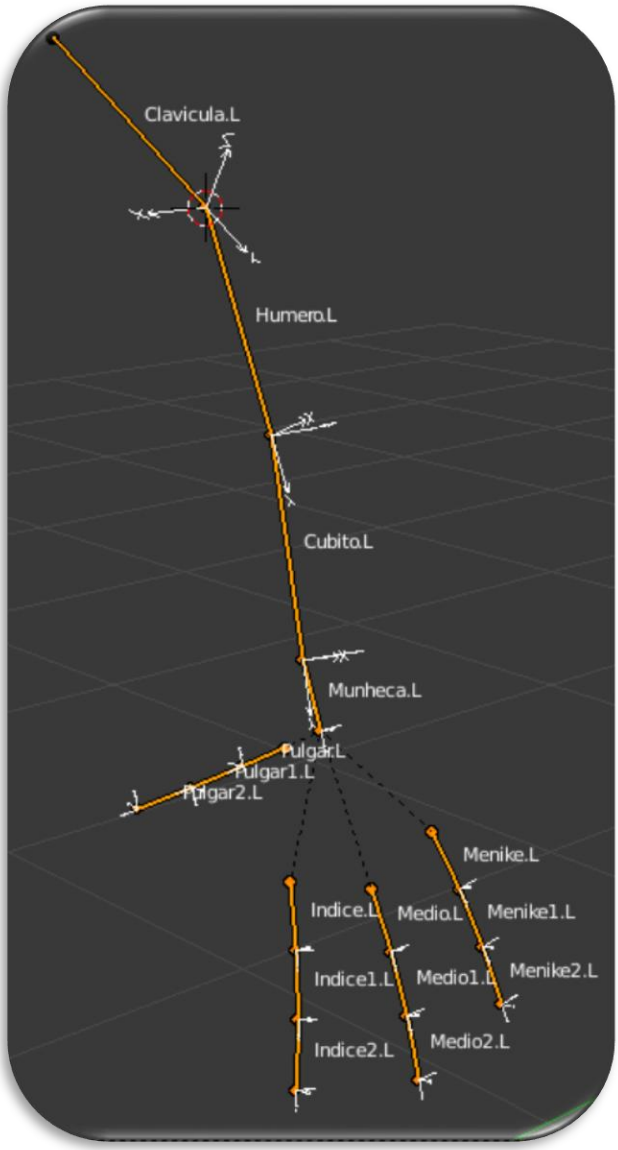


Figura 3 33 Esquema de la conformación de los brazos del robot virtual. En la imagen se visualiza el nombre de cada hueso y su sistema coordenado en el espacio de trabajo de Blender. (Autor).

Tabla 1 Relación parental de los distintos miembros del rigging del brazo Izquierdo de nuestro robot virtual.

HUESO	**PADRE**	**HIJO**
Clavicula.L	Pecho	Humero.L
Humero.L	Clavicula.L	Cubito.L
Cubito.L	Humero.L	Munheca.L CtrlMuneca.L *
Munheca.L	Cubito.L	Pulgar.L Indice.L Medio.L Meñike.L EsclkPulgar.L * EsclKIndice.L EsclkMedio.L * EsclkMeñike.L *
Pulgar.L	Munheca.L	Pulgar1.L
Pulgar1.L	Pulgar.L	Pulgar2.L
Pulgar2.L	Pulgar1.L	--
Indice.L	Munheca.L	Indice1.L
Indice1.L	Indice.L	Indice2.L
Indice2.L	Indice1.L	--
Medio.L	Munheca.L	Medio1.L
Medio1.L	Medio.L	Medio2.L
Medio2.L	Medio1.L	--

Tabla 2 Relación parental de los distintos miembros del rigging del brazo derecho de nuestro robot virtual.

HUESO	**PADRE**	**HIJO**
Clavicula.R	Pecho	Humero.R
Humero.R	Clavicula.R	Cubito.R
Cubito.R	Humero.R	Munheca.R CtrlMuneca.R *
Muñeca.R	Cubito.R	Pulgar.R Indice.R Medio.R Meñike.R EsclkPulgar.R * EsclKIndice.R * EsclkMedio.R * EsclkMeñike.R *
Pulgar.R	Munheca.R	Pulgar1.R
Pulgar1.R	Pulgar.R	Pulgar2.R
Pulgar2.R	Pulgar1.R	--
Indice.R	Munheca.R	Indice1.R
Indice1.R	Indice.R	Indice2.R
Indice2.R	Indice1.R	--
Medio.R	Munheca.R	Medio1.R
Medio1.R	Medio.R	Medio2.R
Medio2.R	Medio1.R	--

Piernas:

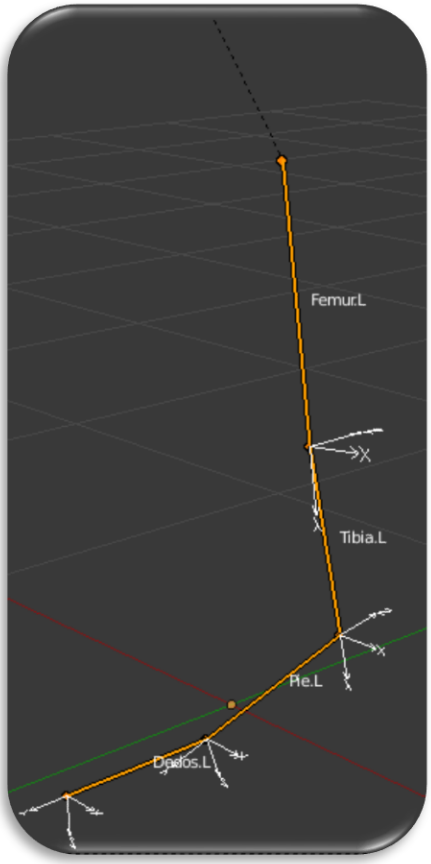


Figura 3 34 Esquema de la conformación de los pies del robot virtual. En la imagen se visualiza el nombre de cada hueso y su sistema coordenado en el espacio de trabajo de Blender. (Autor).

Tabla 3 Relación parental de los distintos miembros del rigging de la pierna Izquierda de nuestro robot virtual.

HUESO	**PADRE**	**HIJO**
Femur.L	Central *	Tibia.L
Tibia.L	Femur.L	Pie.L
Pie.L	Tibia.L	Dedos.L IKPierna.L
Dedos.L	Pie.L	--

Tabla 4Relacion parental de los distintos miembros del rigging de la pierna derecha de nuestro robot virtual.

HUESO	**PADRE**	**HIJO**
Femur.R	Central *	Tibia.R
Tibia.R	Femur.R	Pie.R
Pie.R	Tibia.R	Dedos.R IKPierna.R
Dedos.R	Pie.R	--

Rigging del Tronco, cabeza (y ojos) y cola.

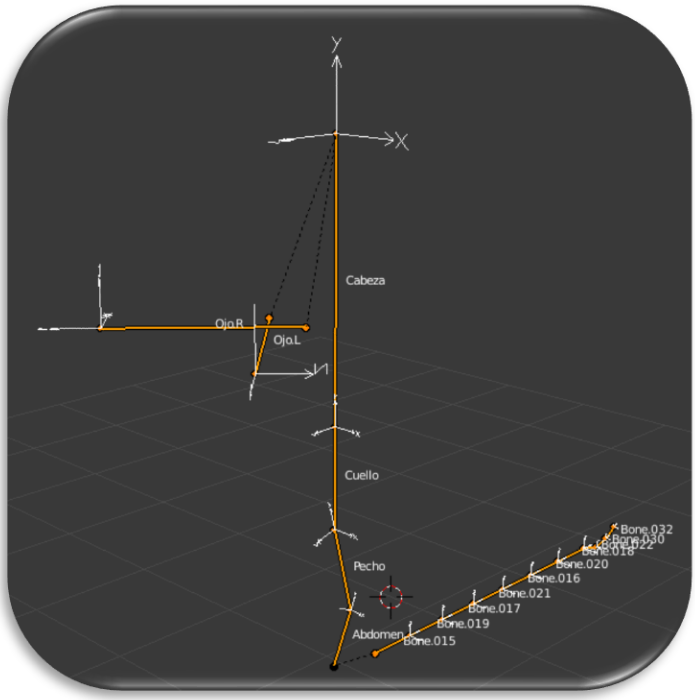


Figura 3 35 Esquema de la conformación de la espina del robot virtual. En la imagen se visualiza el nombre de cada hueso y su sistema coordenado en el espacio de trabajo de Blender. (Autor).

Tabla 5 Relación parental de los distintos miembros del rigging de tronco, cabeza y cola de nuestro robot virtual.

HUESO	**PADRE**	**HIJO**
Abdomen	Central	Pecho
Pecho	Abdomen	Cuello Clavicula.L Clavicula.R
Cuello	Pecho	Cabeza
Cabeza	Cuello	Ojo.L Ojo.R CtrlOjos * Cachete.L Cachete.R CtrlCachete *
Ojo.L	Cabeza	--
Ojo.R	Cabeza	--

Rigging Control del "Eskeleto"

Estos son los miembros del rigging que van a controlar las cinemáticas de las distintas cadenas descritas para la animación, en el caso de que se haga la animación manual. Necesarios, ya que la animación de nuestro robot virtual parte de un proceso mixto, es decir, que combina por una parte la reproducción de los movimientos que se capture con nuestro dispositivo y se complementa en detalle por técnicas manuales.

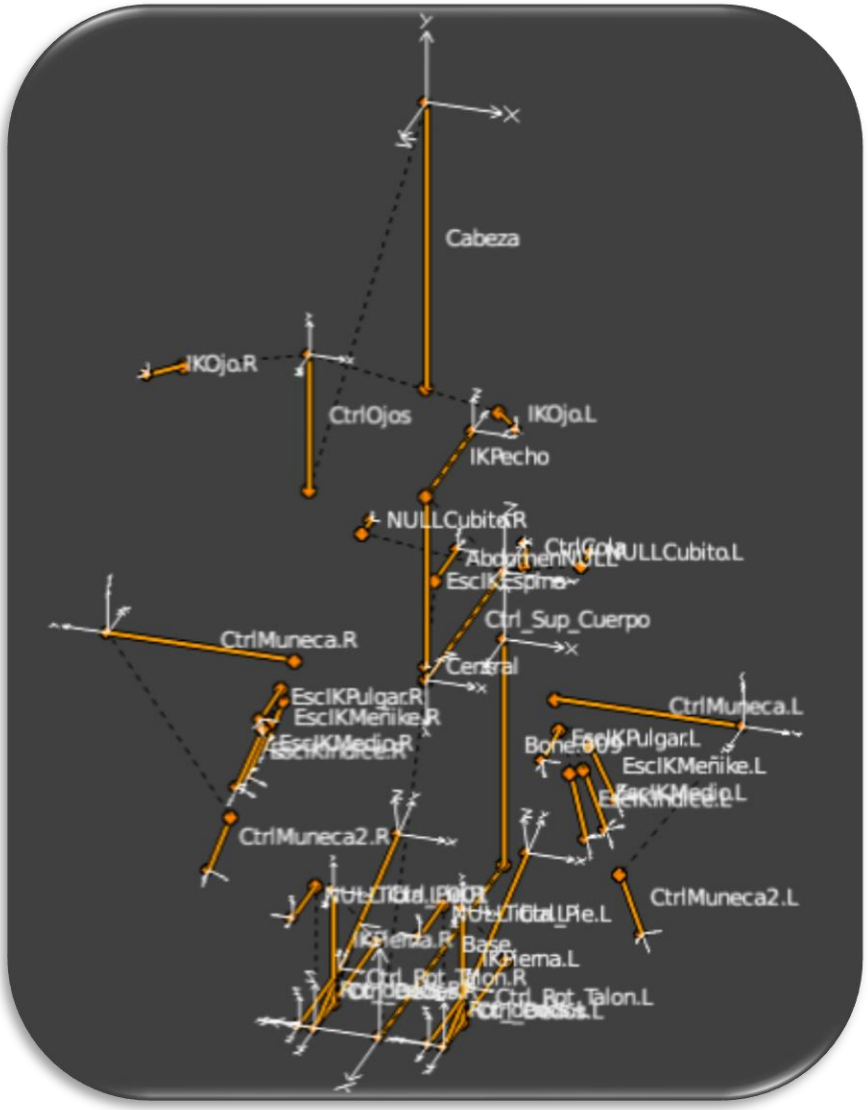


Figura 3 36 Esquema completo de los huesos de control del Robot virtual. (Autor).

Tabla 6 Relación parental de los distintos miembros del rigging de control de nuestro robot virtual.

SECCIÓN	HUESO	PADRE	HIJO
	Base	--	Ctrl_Sup_Cuerpo
	Ctrl_Sup_Cuerpo	Base	Central IKPecho NULLCubito.L – NULLCubito.R – CtrlCola
	Central	Ctrl_Sup_Cuerpo	Abdomen AbdomenNULL Femur.L Femur.R
	IKPecho	Ctrl_Sup_Cuerpo	EsclKEspina
	CtrlOjos	Cabeza *	IKOjo.L IKOjo.R
	CtrlCachete	Cabeza *	IKCachete.L IKCachete.R
Pie Izquierdo	IKPierna.L	Tibia.L	Ctrl_Rot_Talon.L
	Ctrl_Rot_Talon.L	IKPierna.L	Ctrl_Dedos.L
	Ctrl_Dedos.L	Ctrl_Rot_Talon.L	Rot_dedos.L Ctrl_Pie.L IKPie_01.L
	Rot_dedos.L	Ctrl_Dedos.L	IKDedos_01.L
	Ctrl_Pie.L	Ctrl_Dedos.L	IKTibia.L
Pie Derecho	IKPierna.R	Tibia.R	Ctrl_Rot_Talon.R
	Ctrl_Rot_Talon.R	IKPierna.R	Ctrl_Dedos.R
	Ctrl_Dedos.R	Ctrl_Rot_Talon.R	Rot_dedos.R Ctrl_Pie.R IKPie_01.R
	Rot_dedos.R	Ctrl_Dedos.R	IKDedos_01.R
	Ctrl_Pie.R	Ctrl_Dedos.R	IKTibia.R

Mano Izquierda	CtrlMuneca.L	Cubito.L	CtrlMuneca2.L IKCubito.L NULLMuneca.L
	EsclKPulgar.L	Munheca.L	IKPulgar.L NULLPulgar.L
	EsclKIndice.L	Munheca.L	IKIndice.L NULLIndice.L
	EsclKMedio.L	Munheca.L	IKMedio.L NULLMedio.L
	EsclKMeñike.L	Munheca.L	IKMenike.L NULLMenike.L
Mano Derecha	CtrlMuneca.R	Cubito.R	CtrlMuneca2.R IKCubito.R NULLMuneca.R
	EsclKPulgar.R	Munheca.R	IKPulgar.R NULLPulgar.R
	EsclKIndice.R	Munheca.R	IKIndice.R NULLIndice.R
	EsclKMedio.R	Munheca.R	IKMedio.R NULLMedio.R
	EsclKMeñike.R	Munheca.R	IKMenike.R NULLMenike.R

3.3.5. Algoritmo de reproducción de datos.

Para reproducir los datos enviados después de la captura de movimiento, animar el personaje, es necesario escribir un programa en Python que, por medio de un socket UTP dispuesto para comunicarse, reciba una matriz con los ángulos de Euler XYZ de cada hueso y la posición XY del sistema general del esqueleto.

CAPITULO 4.
RESULTADOS

4.1. DESCRIPCIÓN GENERAL DEL SISTEMA DE CAPTURA DE MOVIMIENTO.

Se desarrolló una aplicación con el sensor de profundidad Kinect, el cual es capaz de hacer una captura completa de los 21 puntos somatométricos del cuerpo, lográndose dos objetivos en específico. El primero calculando las relaciones angulares del cuerpo y reproduciéndoselas a un robot virtual, el cual es modelado previamente y mediante la técnica llamada *rigging* (explicada en el primer capítulo), se reproducen estos ángulos en la estructura modificadora denominada *"Eskeleton"*.

El segundo, haciendo un seguimiento frame a frame del punto de muñeca de ambas manos, se calcula por medio de diferenciales de posición y después de haber pasado por un punto determinado, la intención de gesto que se quiera hacer, para luego traducir dicho movimiento en una acción en particular.

Pero para poder realizarlo se crea un escenario de trabajo el cual es adaptable a las necesidades requeridas.

4.1.1. Base del Kinect.

Para establecer la altura del Kinect se ha dispuesto una base como se muestra en la Figura 3.1.6, las especificaciones de la base se encuentran en el **ANEXO 2** para más información. La base es graduable para que tome las alturas distintas de su rango (Hasta 60 cm).

Figura 4 1 Base del Kinect diseñada para el propósito específico. (Autor).

En este capítulo veremos cómo esto se da, tanto para animaciones como para presentaciones de alto impacto.

4.2. ANIMACIONES: DESCRIPCIÓN GENERAL.

Como resultado del desarrollo de este trabajo, se obtuvo un sistema de captura que permite calcular las relaciones angulares del cuerpo de una persona captado desde el sensor Kinect, que permite guardar dichas relaciones en su propio fichero de variables para luego reproducirlas en el modelo virtual desarrollado en Blender.

En Blender se creó un algoritmo capaz de descomponer el archivo generado anteriormente y asignar los valores angulares a las distintas componentes de rotación de las distintas partes del rigging y a su vez asignar el grupo de *KeyFrame* que irán guardando dichas posiciones para luego ser reproducidas en forma de secuencia para su posterior animación. El proceso es el descrito en la ***Figura 4.2.***

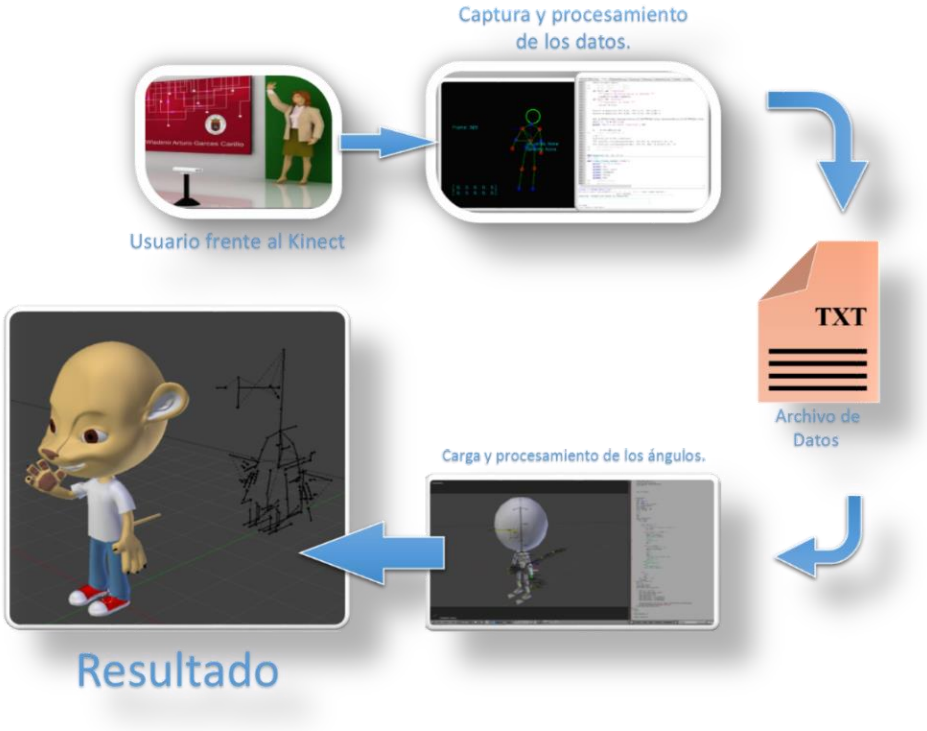


Figura 4 2 Descripción del sistema para animaciones.

En la ***Figura 4.2*** se ve en el sentido de la flecha que el usuario se posiciona delante del dispositivo, luego se recolecta, identifica y procesa la información, luego genera un archivo *txt el cual funciona como un envase de los datos, posteriormente para la reproducción se carga el archivo *txt en el algoritmo que se diseñó para la extracción y copiado de los datos en Blender, sobre el modelo.

4.2.1. Interfaz gráfica.

En los aspectos más precisos la interfaz tendrá una ventana de bienvenida con un botón de *empezar*, una vez presionado este botón el inicializará el Kinect y todas las funciones del programa para su debido uso, la ***Figura 4.3*** da un ejemplo de esto.

Figura 4 3. Ventana de inicio de la interfaz con la opción para empezar.

Posteriormente se abrirá una ventana de funcionalidades. Esta contiene los distintos botones requeridos para poner en marcha el algoritmo, como se muestra en la ***Figura 4.4***.

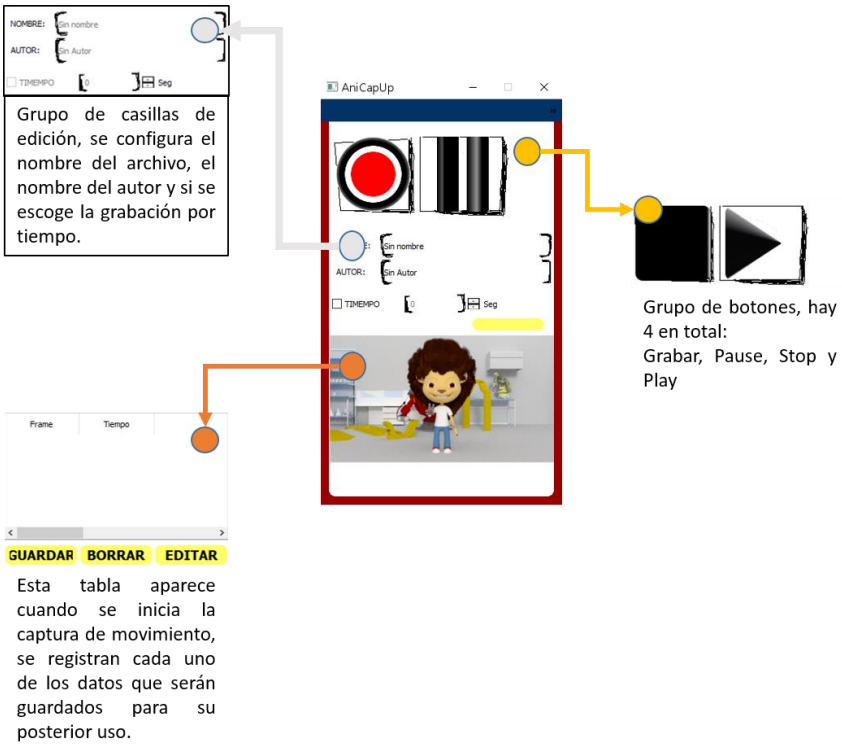


Figura 4 4. Interfaz gráfica para la captura de las relaciones angulares del cuerpo.

En la **Figura 4.4,** se ve la interfaz gráfica para animaciones, donde se ven los botones y los distintos espacios que la componen, la interfaz tiene un acabado informal simulando estar tachada con lápiz en los espacios editables de la aplicación.

Esta interfaz permitirá al usuario, entre otras cosas, grabar una secuencia de relaciones angulares por un tiempo determinado, existiendo dos métodos para esto, una con un tiempo preestablecido y la otra por un tiempo indefinido hasta que el usuario decida si es suficiente, o simplemente haya completado el objetivo que tenía en mente. El usuario de este software no tendrá que ser experto en animación para ejecutarlo, sin embargo, si tendrá que tener conocimientos previos en ello para reproducir las relaciones en el entorno de Blender, por ende, este programa solo es recomendable para personas con conocimientos en animación y edición, como tal.

4.2.2. Descripción de los personajes 3D.

Como se vió anteriormente, usando la herramienta de desarrollo de modelos tridimensionales, Blender 3D, se crearon dos personajes en 3D que serán la insignia clave de este software, no obstante, el programa queda presto para que se reproduzca movimiento en cualquier personaje que la persona desee, teniendo en cuenta algunas restricciones que se mencionarán más adelante. Por el momento el resultado de los personajes es el que se muestra en la ***Figura 4.5,*** donde se muestra claramente las mejoras respecto al modelo que se tenía inicialmente, y en la ***Figura 4.6*** donde se hicieron modificaciones adicionales al modelo original.

Figura 4 5. Resultado del modelo expuesto en secciones anteriores. (Autor).

Figura 4 6. Modificaciones al modelo final.

4.2.3. Reproducción de las relaciones angulares en Blender.

Para la reproducción de las relaciones angulares en Blender se ha dispuesto dos archivos. El primero se presenta en formato **.blend* el cual contiene el rigging nombrado de forma preestablecida, al cual se le van a asignar las relaciones angulares a sus componentes, el otro está en formato **.py* el cual contiene el algoritmo que cargará el archivo que previamente, se genera con el software de captura. El archivo **.py* tendrá que ser cargado en el editor de texto de Blender y posteriormente hacerle la modificación del nombre por el archivo generado previamente en la función definida del programa que sirve para dicho objetivo (*def Cargar_Archivo(self)*), de la manera como se muestra en la imagen (***Figura 4.7***)

Figura 4 7. Programa en Python que se ejecuta desde Blender.

En este programa podemos también cambiar la variable *Key* la cual contiene el intervalo de los fotogramas clave que va a ir tomando, es decir, "Cada cuantos frame's va a haber un fotograma clave", esto con el fin de reducir el ruido creado por el Kinect debido a que los puntos no son precisamente estables, esto se nota con mayor facilidad cuando se establece cada fotograma obtenido como fotograma clave. Por ende, es recomendable establecer un intervalo entre fotogramas clave, de paso se aprovecha que Blender hace interpolaciones entre posiciones, obteniendo movimientos mucho más suaves. Esto se puede comprobar en los documentos digitales anexos a este, haciendo la comparación con los archivos *Prueba1.blend y Prueba2.blend,* en los cuales, en uno se toman intervalos entre fotogramas clave de 1 y de 6 respectivamente, y que corresponden a un intervalo del 0% y 20% entre grupo de datos por cada segundo (o cada 30 frame's). ***Ver Figura 4.8 y Figura 4.9.*** Para estas se tomo un intervalo definido de una secuencia cargada anteriormente (Para este caso frame's del 85 al 100 de la secuencia "Sustentacion.txt" del archivo \Pruebas_animaciones en ANEXOS), se nota que hay cambios en las posiciones de las extremidades en algunos de los frames,

Figura 4 8 Secuencia de imágenes del archivo Prueba1.blend con el archivo "Sustentacion.txt" cargado, esta representa un intervalo del 0% de los datos tomados. (Autor)

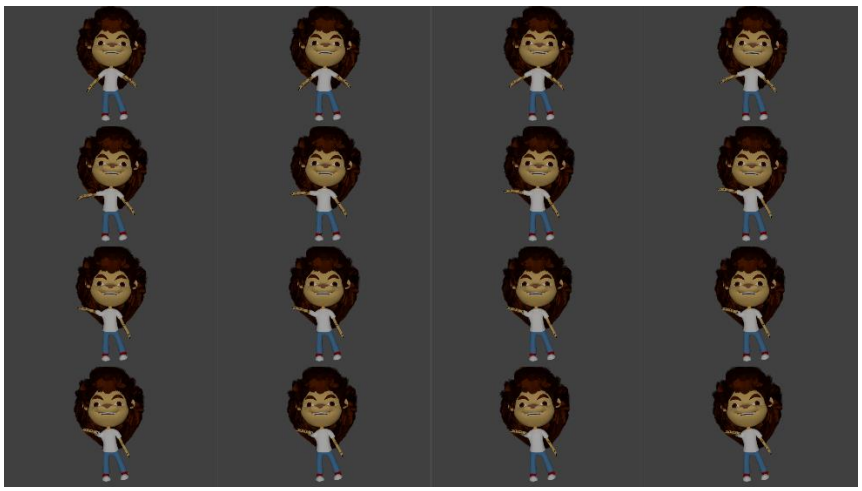


Figura 4 9 Secuencia de imágenes del archivo Prueba2.blend con el archivo "Sustentacion.txt" cargado, esta representa un intervalo del 20% de los datos tomados. (autor)

Figura 4 10 Secuencia de imágenes del archivo Prueba2.blend con el archivo "Sustentacion.txt" cargado, esta representa un intervalo del 50% de los datos tomados. (autor)

> ***NOTA 1:*** *Es recomendable que el intervalo entre fotogramas clave no sea mayor al número de fotogramas obtenidos por segundo (30 frames) ya que se pueden perder posiciones que pueden ser importantes para nuestro uso. Comprobable en el archivo digital Prueba5.blend.*

Habiendo alcanzado los pasos anteriores, entonces, se procede a los detalles que se requieran para nuestra animación, como: escenario, vestimenta, texturizado, etc.

> ***NOTA 2:*** *Se reitera al lector que se debe tener conocimiento previo del manejo del manejo de Blender y/o tener conocimientos generales de animación por computador.*

4.3. PRESENTACIONES: DESCRIPCIÓN GENERAL.

Otro resultado del desarrollo de este trabajo arrojó un sistema de captura de movimiento, que permite hacer el seguimiento constante de los puntos somatométricos de las muñecas captados desde el sensor Kinect (Esto porque es un punto más estable que el de la palma de la mano), esperando que pase por la posición determinada llamada *Static_X* para posteriormente recolectar un vector de posiciones durante el siguiente segundo, dicho vector es luego usado para determinar mediante los diferenciales de las posiciones en cada eje, un gesto. Este gesto es usado para hacer el control del apoyo visual para nuestras Presentaciones de Alto Impacto (diapositivas).

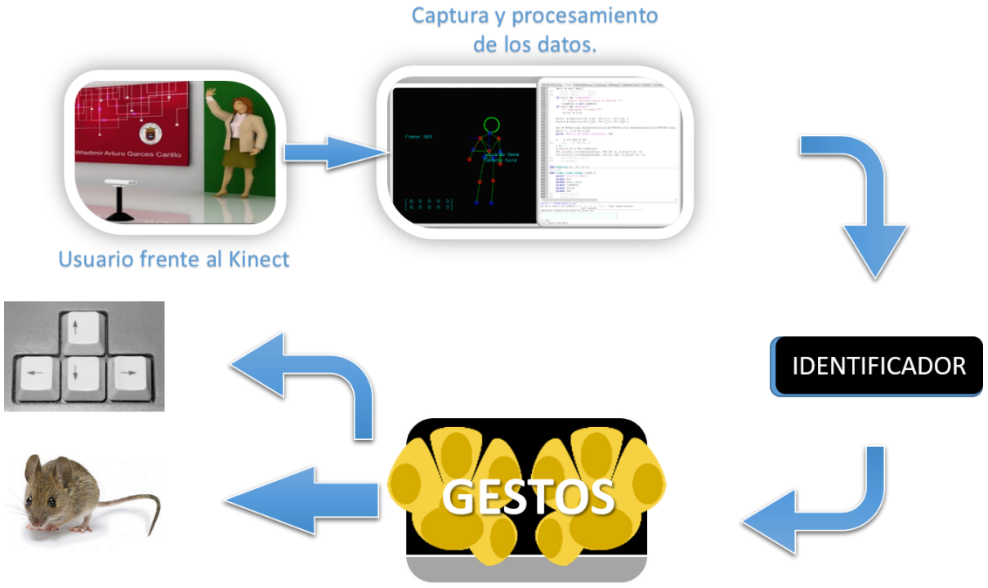


Figura 4 11 Esquema del programa que se desarrolló.

En la ***Figura 4.8*** se ve en el sentido de la flecha que el usuario se posiciona delante del dispositivo, luego se recolecta y procesa la información, después identifica cual fue el movimiento que se produjo, lo traduce en un gesto para generar, por último, un evento de *mouse* o *teclado* sobre la presentación que tenemos abierta.

Además de desarrollar esta herramienta tecnológica, se diseñó un módulo metodológico para la implementación de presentaciones de alto impacto el cual se describe en los anexos (**ANEXO 3**) y que potencializa el uso de nuestra herramienta.

4.3.1. Descripción de la Interfaz.

La interfaz es relativamente sencilla y compacta. Al igual que la interfaz de animaciones, esta contiene una ventana de bienvenida **(Figura 4.3)**. Esta permite al usuario cargar una diapositiva al sistema, sobre la que se va a hacer el control. Cuenta con visualizadores de apoyo para los usuarios, que incluye la visualización de cada gesto, como la visualización de las diferentes cámaras del Kinect, con el objetivo de proveer al nuevo usuario de apoyo y aprendizaje del uso de nuestro sistema.

La interfaz, en la **Figura 4.9**, se ejecuta con una guía de comprobación que permite al sistema ajustar sus valores respecto a las medidas corporales del individuo que está dispuesto frente a él. Además, cuenta con botones de control del ángulo de inclinación del Kinect y de su altura. Así como, su menú de acceso a la documentación y al cargador de las diapositivas, por último, cuenta con un botón de inicio de nuestro programa, por si se quiere hacer manual el inicio del mismo.

Figura 4 12 Interfaz gráfica para la captura y reproducción de gestos.

4.3.2. Descripción de los gestos y su respectivo comando.

Los gestos serán descritos de la siguiente forma, ***ver Tabla 7***:

Tabla 7 Estados de los gestos de las dos manos.

GESTO DERECHO				GESTO IZQUIERDO					
D	***I***	***A***	***Ab***	***D***	***I***	***A***	***Ab***	**ACCIÓN**	**Send Keys**
0	0	0	0	0	0	0	0	Nada	
1	0	0	0	0	0	0	0	Diapositiva a la Izquierda.	←: "{LEFT}"
0	1	0	0	0	0	0	0	Diapositiva a la Derecha.	→: "{RIGHT}"
0	0	1	0	0	0	0	0	Abrir pantalla completa 1.	F5': "{F5}"
0	0	0	1	0	0	0	0	START.	
0	0	0	0	1	0	0	0	Diapositiva a la Izquierda.	←: "{LEFT}"
0	0	0	0	0	1	0	0	Salir de pantalla.	Esc: "{ESC}"
0	0	0	0	0	0	1	0	Mostrar RGB y ESK.	
0	0	0	0	0	0	0	1	Ocultar RGB y ESK.	
1	0	0	0	0	1	0	0	Abrir pantalla completa 2.	Alt+'F5':"%{F5}"
0	1	0	0	1	0	0	0	Control de Mouse.	
0	0	1	0	0	0	1	0	Abrir pantalla completa 3.	F5': "{F5}"
0	0	0	1	0	0	0	1		

Donde ***D:*** *Derecha,* ***I:*** *Izquierda,* ***A:*** *Arriba &* ***Ab:*** *Abajo.*

4.3.3. Pruebas.

Para determinar la efectividad de nuestra herramienta tecnológica se hicieron varias pruebas en dos instantes de tiempo diferentes y con poblaciones distintas, desbordando estas en 4 encuestas.

Descripción de las pruebas.

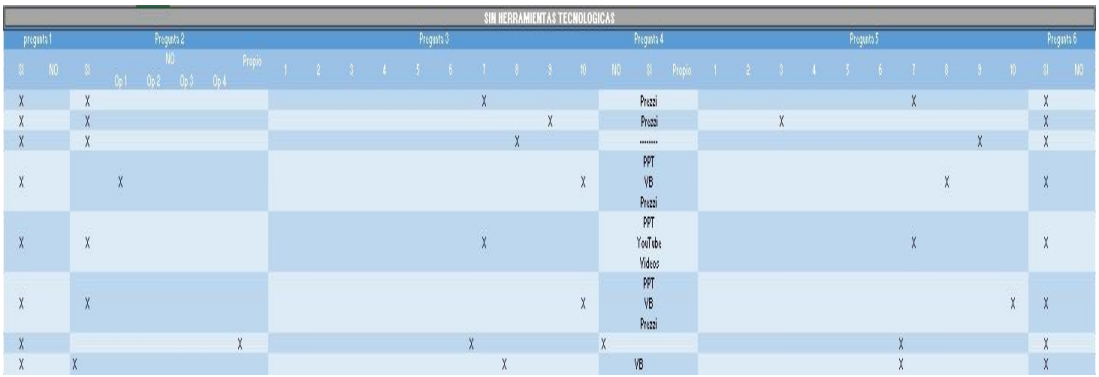
La prueba constaba en hacer una presentación corta describiendo la metodología para presentaciones de alto impacto descrita en los anexos a profesores de distintas áreas del conocimiento en el período inter-semestral de los períodos académicos del año 2015-I y 2015-II, reunidos para capacitación por PLANESTIC en distintas aulas al interior de la Universidad de Pamplona, incluida el AULATIC de la Facultad de Ingenierías. Las pruebas se hicieron de la siguiente forma:

Prueba 1: Sin herramientas tecnológica. (Profesores)

Esta prueba se realizó en el salón J200 de la biblioteca de las instalaciones de la Universidad. Esta prueba se realiza sin ningún tipo de recurso o herramienta

tecnológica arrojando los resultados en la tabla. Se nota que la totalidad de personas encuestadas encontraron útil la información suministrada, que la mayoría de las personas usan algún tipo de metodología para hacer presentaciones sin embargo el 25% de los encuestados en esta prueba desconocen alguna metodología para el desarrollo de presentaciones efectivas; también evidencia que en promedio la información del módulo expuesto obtuvo una puntuación de 8,125 puntos que en consideración de estos resultados se considera buena. La mayoría de las personas encuestadas usan herramientas visuales para el soporte de sus presentaciones siendo Prezzi, Power Point, implementados por medio de VideoBeam, los mas comunes y usados por los encuestados.

Tabla 8 Datos de la primera prueba.



SIN HERRAMIENTAS TECNOLOGICAS																																					
pregunta 1		Pregunta 2						Pregunta 3										Pregunta 4			Pregunta 5										Pregunta 6					
SI	NO	SI	NO				Propio	1	2	3	4	5	6	7	8	9	10	NO	SI	Propio	1	2	3	4	5	6	7	8	9	10	SI	NO				
			Op 1	Op 2	Op 3	Op 4																															
X		X												X					Prezzi							X				X						
X		X														X			Prezzi			X								X						
X		X													X				.....										X		X						
X			X														X		PPT VB Prezzi								X			X						
X		X												X					PPT YouTube Videos							X				X						
X		X															X		PPT VB Prezzi										X	X						
X						X								X				X								X				X						
X		X													X				VB							X				X						

Prueba 2: Con herramientas tecnológicas parciales. (Profesores)

Esta prueba se realizó en las instalaciones del ICDL, al interior de la universidad. Se usaron para la prueba un puntero inalámbrico y videobeam para la presentación del módulo descrito en los anexos arrojando los resultados de la tabla. Se nota que la totalidad de personas encuestadas encontraron útil la información suministrada, que la mayoría de las personas usan algún tipo de metodología para hacer presentaciones sin embargo el 16% de los encuestados en esta prueba desconocen alguna metodología para el desarrollo de presentaciones efectivas; también evidencia que en promedio la información del módulo expuesto obtuvo una puntuación de 9,16 puntos que en consideración de estos resultados se considera *muy buena*. La mayoría de las personas encuestadas usan herramientas visuales para el soporte de sus presentaciones siendo el uso de VideoBeam la herramienta tecnológica más común y usada por los encuestados.

Tabla 9 Datos de la segunda prueba.

HERRAMIENTAS TECNOLOGICAS PARCIALES																																
pregunta 1		Pregunta 2						Pregunta 3										Pregunta 4			Pregunta 5										Pregunta 6	
SI	NO	SI	NO Op 1	NO Op 2	NO Op 3	NO Op 4	Propio	1	2	3	4	5	6	7	8	9	10	NO	SI	Propio	1	2	3	4	5	6	7	8	9	10	SI	NO
X							X										X		PPTX Prezzi VB											X	X	
X		X														X				Mouse										X	X	
X		X															X		VB Mouse											X	X	
X		X													X				CD										X		X	
X		X														X			----										X		X	
X		X														X			VB CD										X		X	

Prueba 3: Con la herramienta tecnológica desarrollada. (Profesores)

Esta prueba se realizó en el AulaTIC de la Facultad de Ingenierías y Arquitectura, en las instalaciones de la universidad. Se usaron para la prueba los recursos dispuestos en el AulaTIC además de la herramienta descrita en este documento para la presentación del módulo descrito en los anexos arrojando los resultados de la tabla. Se nota que la totalidad de personas encuestadas encontraron útil la información suministrada, que la mayoría de las personas usan algún tipo de metodología para hacer presentaciones efectivas; también evidencia que en promedio la información del módulo expuesto obtuvo una puntuación de 7,6 puntos que en consideración de estos resultados se considera *bueno*. La mayoría de las personas encuestadas usan herramientas visuales para el soporte de sus presentaciones siendo el uso de VideoBeam la herramienta tecnológica más común y usada por los encuestados.

Tabla 10 Datos de la prueba 3

CON LA HERRAMIENTA TECNOLOGICA DESARROLLADA																																
pregunta 1		Pregunta 2						Pregunta 3										Pregunta 4			Pregunta 5										Pregunta 6	
SI	NO	SI	NO Op 1	NO Op 2	NO Op 3	NO Op 4	Propio	1	2	3	4	5	6	7	8	9	10	NO	SI	Propio	1	2	3	4	5	6	7	8	9	10	SI	NO
X		X														X			Emmaze Prezzi											X	X	
X		X													X				Prezzi VB								X				X	
X		X											X					X										X			X	
X		X														X			PPT VB Prezzi								X				X	
X		X													X				PPT YouTube Videos											X	X	
X		X															X		PPT VB Prezzi											X	X	
X		X														X			VB										X		X	
X		X															X		VB											X	X	
X		X													X				--------									X			X	
X		X														X			PPT YouTube Videos									X			X	

Prueba 4: Evaluación de usuario de la herramienta tecnológica desarrollada. (Indistinto).

Esta prueba se realizó en varios escenarios. Se trató de hacer un análisis de como las personas usaban el sistema. Para esta prueba se sometió el sistema a evaluación de personas de distintas áreas de conocimiento (***Ver Figura 4.10)***.

Figura 4 13 Estudiante de psicología posado en frente del sistema empezando a usarlo.

Quienes, con previo entrenamiento con la herramienta, evaluaron el sistema. En la ***Figura 4.11*** se ve una secuencia de uno de los gestos en ejecución.

Figura 4 14 Secuencia del gesto realizado con la mano derecha desplazándose a la izquierda y su efecto en la diapositiva proyectada en el fondo de la imagen.

CAPITULO 5.
CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES.

- La técnica implementada para el modelado de los personajes de la Universidad de Pamplona es relativamente sencilla. Las curvas tipo *Bezier* simplifican el entendimiento de las formas principales del modelo. Sin embargo, se deben complementar con la ubicación manual de los planos (Mediante extrusión de ejes) que conforman la malla de los modelos, debido a que solo las curvas no representan superficies completas. Por lo cual, aunque goza el modelado de una sencillez particular, requiere alto costo temporal.
- Los *riggings* de las articulaciones de los personajes de la Universidad de Pamplona que se construyeron cuentan con sistemas de control distintos. Se encontró que el control cinemático completo con que cuenta uno de ellos arroja buenos resultados en cuanto a la fluidez del movimiento, sin embargo, las horas de desarrollo que se tienen que emplear son muchas y requieren de un equipo de trabajo amplio para secuencias animadas complejas.
- El uso del paradigma Orientado a Objeto y Eventos, POO y POE respectivamente, así como la Programación en Paralelo mediante el uso de hilos para la evolución, desarrollo y optimización de la solución del problema, arrojó mejores resultados que mediante el paradigma convencional (Programación Estructurada), así se alcanzó con mayor precisión el objetivo de poder procesar en el menor tiempo posible el volumen de datos de la resolución máxima del Kinect; es decir, que de los 30 grupos de datos que arroja el Kinect cada segundo, se pudo usar cada uno de ellos, dando la ilusión de hacerlo en tiempo real, para poderlos reproducir a los modelos posteriormente.
- La estructura que se usó para registrar los datos capturados fue sencilla y no represento un costo computacional elevado ya que se hace por medio de la generación de un archivo de texto convencional de extensión “*.txt”, ya que es compatible con múltiples plataformas.
- La herramienta Blender se presenta como un entorno viable para el desarrollo de estructuras robóticas virtuales, ya que esta permite la creación, modificación y control de estructuras cinemáticas complejas. También cuenta con su propio compilador de Python donde se escribe el algoritmo de extracción de las relaciones angulares de cada componente, y que en específico se usaron en este trabajo para el control de las manos y los pies del personaje en 3D.
- El control de los *riggings* de las articulaciones de los personajes de la Universidad de Pamplona que se usó mediante la reproducción de los datos recolectados arrojó mejores resultados cuando se usó solo un porcentaje de ellos, debido a los errores intrínsecos de los datos.

- La herramienta desarrollada reduce el estándar de configuraciones de cámaras y dispositivos para la implementación de técnicas MoCap, sin embargo, se necesitaría una configuración más compleja de sensores Kinect para tomar una mejor descripción de los movimientos de un individuo que se pose frente al sistema.
- Las técnicas y paradigmas que se usaron para la recolección de datos en las animaciones obtuvieron resultados óptimos para el control de presentaciones.
- La recolección de los *vectores ventana* que relacionan el seguimiento de los tres puntos del cuerpo que se usaron, facilitaron el cálculo y procesamiento de los gestos descritos en este trabajo.
- La normalización de los datos del *vector ventana* recolectado para el seguimiento del punto de la espina alcanzó la estabilidad de las trayectorias descritas por la muñeca para el posterior cálculo de los gestos, mejorando el efecto de estos sobre el control de las diapositivas.
- Debido al uso de las librerías nativas del sistema operativo Windows (Win32api), se logra, además del control de diapositivas, el control de eventos del mouse y teclado que pueden ser usados para el control del computador en términos generales y que pueden servir para trabajos futuros o continuación de este mismo.
- No puede describir rotaciones de 180° con certeza, es decir que los sujetos solo tienen un rango de rotación sobre el eje “y” entre -80° y 80°, sin embargo, con la adaptación de otros dispositivos y una debida relación entre estos, se podrían obtener mejores resultados; aun así, seguiría siendo un dispositivo bastante económico.

5.2. RECOMENDACIONES.

- Disponer solo un sujeto frente al Kinect.
- No usar ropa con materiales que absorban la luz infrarroja, esto no permitiría que el Kinect hiciera su procesamiento debido.
- Cuando se esté utilizando el sistema para animaciones se debe tratar de no hacer tomas con giros que no estén dentro del rango de los -80° y 80°, debido a que el sistema no puede tomar giros más allá del rango establecido y el resultado no sería acorde con la toma que desea aplicar.
- Cuando se esté utilizando el sistema para presentaciones, no se deben hacer los gestos con el cuerpo girado en el rango establecido anteriormente, debido a que esto generaría errores en el sistema, causando que se descontrole o que tome gestos que no se quieran o no se hayan ejecutado.

REFERENCIAS BIBLIOGRÁFICAS

[1]. Alexiadis, D. S., & Daras, P. (2014). Quaternionic Signal Processing Techniques for Automatic Evaluation of Dance Performances From MoCap Data. *IEEE TRANSACTIONS ON MULTIMEDIA, VOL 16, NO. 5*, 1391-1406.

[2]. Ayala, N. A., Mendívil, E. G., Salinas, P., & Rios, H. (2013). Kinesthetic Learning Applied to Mathematics Using Kinect. *Procedia Computer Science 25*, 131-135.

[3]. Barnachon, M., Bouakaz, S., Boufama, B., & Guillou, E. (2013). A real-time system for motion retrieval and interpretation. *Pattern Recognition Letters 34*, 1789-1798.

[4]. Bonnechére, B., Jansen, B., Salvia, P., Bouzahouene, H., Omelina, L., Moiseev, F., . . . Jan, S. V. (2014). Validity and reliability of the Kinect within functional assessment. *Gait & Posture 39*, 593-598.

[5]. Cassola, F., Morgado, L., Carvalho, F. d., Paredes, H., Fonseca, B., & Martinsa, P. (2014). Online-Gym: a 3D virtual gymnasium using Kinect interaction. *Procedia Technology 13* , 130-138.

[6]. Chang, Y.-J., Han, W.-Y., & Tsai, Y.-C. (2013). A Kinect-based upper limb rehabilitation system to assist. *Research in Developmental Disabilities 34*, 3654-3659.

[7]. Clark, R. A., Pua, Y.-H., Fortin, K., Ritchie, C., Webster, K. E., Denehy, L., & Bryant, A. L. (2012). Validity of the Microsoft Kinect for assessment of postural control. *Gait & Posture 36*, 372-377.

[8]. Du, G., & Zhang, P. (2014). Markerless human–robot interface for dual robot manipulators using Kinect sensor. *Robotics and Computer-Integrated Manufacturing* , 150-159.

[9]. Dutta, T. (2012). *Applied Ergonomics 43*, 645-649.

[10]. Galna, B., Barry, G., Jackson, D., Mhiripiri, D., Olivier, P., & Rochester, L. (2014). Accuracy of the Microsoft Kinect sensor for measuring movement in people with Parkinson's disease. *Gait & Posture 39*, 1062-1068.

[11]. González-Ortega, D., Díaz-Pernas, F., Martínez-Zarzuela, M., & Antón-Rodríguez, M. (2014). A Kinect-based system for cognitive rehabilitation exercises monitoring. *computer methods and programs in biomedicine 113*, 620-631.

[12]. Güdükbay, U., Demir, ˙., & Dedeoglu, Y. (2013). Motioncapture and human pose reconstruction from a single-view video sequence. *DigitalSignalProcessing*, 1441-1450.

[13]. Han, S., & Lee, S. (2013). A vision-based motion capture and recognition framework for behavior-based safety management. *Automation in Construction 35*, 131-141.

[14]. Henseler, H., Kuznetsova, A., Vogt, P., & Rosenhahn, B. (2014). Validation of the Kinect device as a new portable imaging system for three-dimensional breast assessment. *Journal of Plastic, Reconstructive & Aesthetic Surgery 67*, 483-488.

[15]. Ibañez, R., Soria, Á., Teyseyre, A., & Campo, M. (2014). Easy gesture recognition for Kinect. *Advances in Engineering Software*, 171-180.

[16]. Jiang, B., Zhao, F., & Liu, X. (2014). Observation-oriented silhouette-aware fast full body tracking. *Journal of Manufacturing Systems 33*, 209-217.

[17]. Kitagawa, M., & Windsor, B. (2008). *MoCap for Artists: Workflow and Techniques for Motion Capture.* AMSTERDAM, NEW YORK: Focal Press.

[18]. Sandau, M., Koblaucha, H., Moeslund, T. B., Aanæs, H., Alkjær, T., & Simonsen, E. B. (2014). Markerless motion capture can provide reliable 3D gait kinematics in the sagittal and frontal plane. *Medical Engineering & Physics*, 1168-1175.

[19]. Unzueta, L., Peinado, M., Boulic, R., & Suescun, Á. (2008). Full-body performance animation with Sequential Inverse Kinematics. *Graphical Models 70*, 87-104.

[20]. Zhao, X., Li, X., Pang, C., & Wang, S. (2013). Human action recognition based on semi-supervised discriminant analysis with global constraint. *Neurocomputing 105*, 45-50.

ANEXO 1 (Kinect)

Tomado de: (VELASCO ERAZO, 20113)

Kinect es un dispositivo desarrollado por PrimeSense y distribuido por Microsoft para la videoconsola Xbox 360. Su apariencia se muestra en la Figura 1.10. Kinect es un dispositivo de control por movimiento creado originalmente para jugar sin necesidad de ningún mando o controlador para interactuar con el cuerpo en el videojuego mediante una interfaz natural de usuario que reconoce gestos, comandos de voz e imágenes.

Figura 1.10: Sensor Kinect para la consola Xbox360[1]

El lanzamiento del sensor Kinect en noviembre del 2010 supondría una revolución en el uso de este tipo de dispositivos capaces de capturar la profundidad. Su bajo coste y su aceptable precisión en los datos entregados hicieron que pronto éste fuese utilizado en multitud de aplicaciones.

Todos los píxeles que Kinect recibe como IR son convertidos en una escala de colores, haciendo que los cuerpos, dependiendo de la distancia, se capturen como rojos, verdes, azules hasta llegar a tonos grises, que representan a objetos muy lejanos[2].

1.4.1 Partes fundamentales del Hardware

Kinect es una barra de plástico negro de 30 cm de ancho conectada a un cable que se bifurca en dos, un cable USB y otro un cable eléctrico. Se pueden distinguir cuatro partes fundamentales dentro del hardware:

- Cámara RGB, cámara de video con una resolución de 640x480 y 1280x1024 píxeles a 30 fps.

[1] "Kinect". [En línea]. Recuperado el 09 de marzo del 2013, Disponible en Web: <http://www.respuestario.com/como/como-funciona-kinect-guia-de-uso-practica-paso-a-pasor>

[2] "Conoce con detalles técnicos cómo funciona Kinect". [En línea]. Recuperado el 22 de septiembre del 2012, Disponible en Web: <http://www.tierragamer.com/index.php/conoce-como-funciona-kinect/>

- Sensores 3D de profundidad, combinación de un proyector de profundidad con un sensor de profundidad.
- Inclinación monitorizada, permite ajustar la cámara hacia arriba o abajo hasta 27°.
- Micrófono Multi-array, conjunto de cuatro micrófonos que se monta como un solo micrófono.

En la Figura 1.11 se puede distinguir las partes del Kinect.

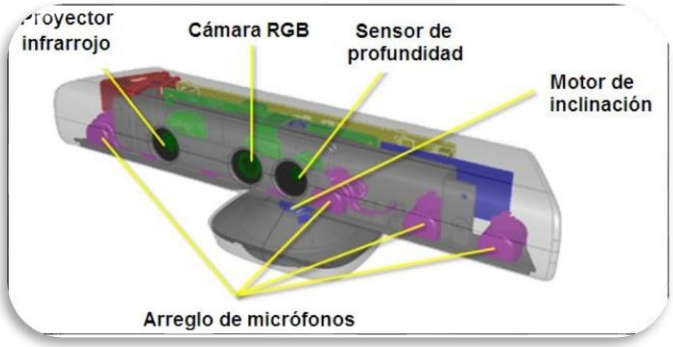


Figura 1.11: Hardware del sensor Kinect[3]

Y aunque no visibles a simple vista, Kinect también posee:

- Memoria RAM de 512 Mb.
- Acelerómetro, para estabilizar la imagen cuando se mueve.
- Ventilador, no está encendido continuamente para no interferir con los micrófonos.

Las especificaciones del sensor Kinect se resumen en la Tabla 1.1.

Se debe inclinar el Kinect pocas veces como sea posible, para minimizar el desgaste en el sensor y para minimizar el tiempo de inclinación. El motor de inclinación no está diseñado para el movimiento constante o repetitivo, y los intentos de utilizarlo de esa manera pueden causar la degradación de la función motora.

Para reducir el desgaste, la aplicación debe cambiar el ángulo de elevación no más de una vez por segundo. Además, debe permitir por lo menos 20 segundos de descanso después de

[3] "Sensor Kinect". [En línea]. Recuperado el 10 de marzo del 2013, Disponible en Web: <http://msdn.microsoft.com/en-us/library/jj131033.aspx>

15 cambios consecutivos. Si se exceden estos límites, el motor de inclinación puede experimentar un período de bloqueo y se traducirá en un código de error.

Tabla 1.1: Especificaciones del Kinect

Kinect	Especificación
Ángulo de visión	43° vertical por 57° horizontal
Rango de inclinación vertical	±27°
Resolución Cámara RBG	640x480(VGA[4]) y 1280X1024 Píxeles
Resolución Sensor de Profundidad	640x480(VGA), 320x240 (QVGA) y 80x60 píxeles
Velocidad de generación	30 imágenes por segundo (30fps)

En la imagen de la cámara RBG el valor por defecto es InfraredResolution640x480Fps30, que son datos de 16 bits, cuya resolución es de 640 x 480 y la velocidad de fotogramas es de 30 fotogramas por segundo.

1.4.2 Técnica para capturar la profundidad

La cámara permite generar una imagen tridimensional de lo que tiene delante y además reconocer partes del cuerpo humano. Para ello utiliza un sónar de luz infrarroja.

El proyector láser infrarrojo proyecta sobre la escena un patrón de 50000 puntos invisibles al ojo humano, como se muestra en la Figura 1.12 (la fotografía fue obtenida con una cámara de visión nocturna), luego de rebotar en los objetos de la escena el patrón de puntos es captado por la cámara infrarroja que se encuentra a 7,5cm de separación del proyector.

Entonces el circuito integrado, analiza la disparidad provocada por los objetos de la escena entre el patrón de puntos proyectados y el patrón de puntos captados[5].

[4] El formato VGA (Colección de Gráficos de Video) fue diseñado por IBM en 1987 con un tamaño de imagen de 640x480. Debido a estándares comunes para PCs y monitores industriales, VGA ha sido ampliamente usado en los dispositivos con imágenes digitales.

[5] MESÍAS, Alejandro; LÓPEZ, Christian. “Diseño e implementación de un prototipo a escala de un robot móvil acompañante”. Director: Nelson Sotomayor MSc. Escuela Politécnica Nacional, Facultad de Eléctrica y Electrónica. Quito, Junio 2012.

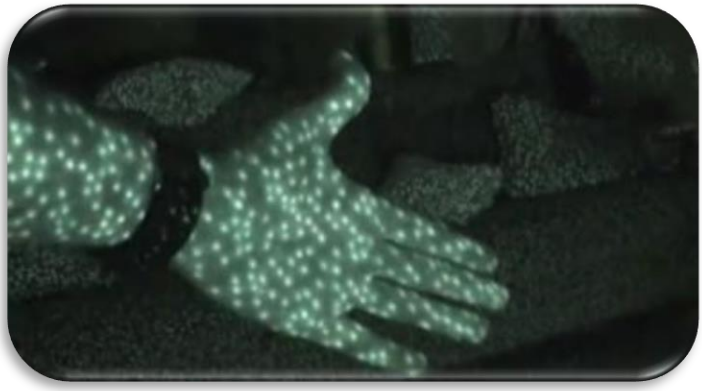


Figura 1.12: Proyección del patrón infrarrojo[6]

Cuando la cámara recibe la luz infrarroja generada por el cañón de infrarrojos se combina con el sensor monocromático CMOS se genera una malla de puntos mediante los cuales se genera una imagen como la Figura 1.13:

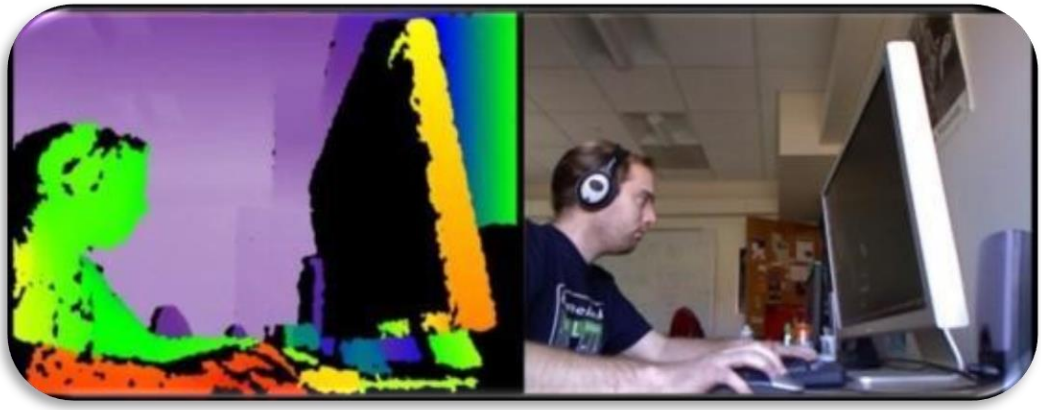


Figura 1.13: Imagen generada por la cámara del Kinect[7]

El entorno que la cámara enfoca, así como la profundidad de los objetos es de la que se parte para un procesado más completo. Acto seguido el chip de procesado de imagen (PrimeSense PS1080) descompone la imagen en los parámetros necesarios para su tratamiento. El proceso se resume en la Figura 1.14.

[6] "Proyector láser infrarrojo". [En línea]. Recuperado el 1 de octubre del 2012, Disponible en Web: < http://biqfr.blogspot.com/2010_11_02_archive.html >

[7] "Programar sensor Kinect". [En línea]. Recuperado el 1 de octubre del 2012, Disponible en Web: <http://biqfr.blogspot.com/2010_11_02_archive.html>

Figura 1.14: Diagrama de la tecnología Kinect patentada por PrimerSense[8]

1.4.3 Información de profundidad

La información de profundidad se devuelve en un mapa de píxeles con una frecuencia máxima de 30 imágenes por segundo. Cada pixel está representado por dos bytes (16 bits), cuyo valor representa teóricamente la distancia del objeto al sensor.

Si el valor de un pixel del mapa es cero, significa que el sensor no pudo estimar la profundidad para esa región debido a las limitaciones del sensor.

Cada píxel contiene la distancia cartesiana, en milímetros, desde el plano de la cámara al objeto más cercano en esa coordenada particular (x,y) como se muestra en la Figura 1.15. Las coordenadas (x,y) de la imagen de profundidad no representan unidades físicas en la habitación, sino que representan la ubicación de un píxel en la imagen de profundidad.

Los datos de profundidad real son de un tamaño de 12 bits. Los 2 bytes de datos de profundidad incluyen un índice de jugadores en los tres bits menos significativos. Los datos de profundidad, por lo tanto, están desplazados a la izquierda por 3 bits. Se debe tener esto

[8] "PrimerSense", [En línea]. Recuperado el 1 de octubre del 2012. Disponible en Web: <http://www.primesense.com/en/technology/115-the-primesense-3d-sensing-solution>

en cuenta a la hora de leer los bits de profundidad[9]. En la Figura 1.16 se muestra la localización de los datos de profundidad.

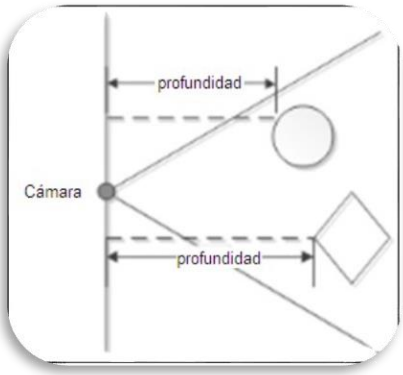


Figura 1.15: Forma de medir la profundidad del Kinect[10]

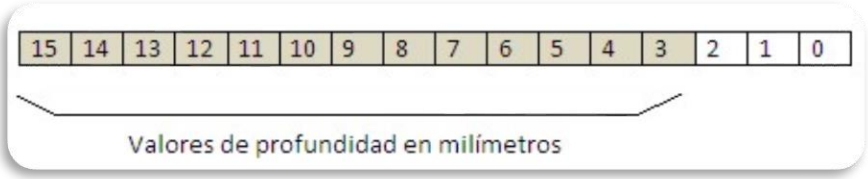


Figura 1.16: Ubicación de los bits de profundidad

Hay tres valores que indican que la profundidad no podría determinarse de forma fiable en una ubicación. El valor "demasiado cerca" significa que un objeto fue detectado, pero está muy cerca del sensor para proporcionar una medición de distancia fiable. El valor "demasiado lejos" significa que un objeto fue detectado, pero está muy lejos poder medirse de forma fiable. El valor "desconocido" significa que ningún objeto fue detectado.

El sensor de profundidad tiene dos rangos de profundidad: el rango por defecto y el rango cercano. La Figura 1.17 ilustra los rangos de profundidad del sensor en metros. El rango predeterminado está disponible tanto en el sensor Kinect para Windows y para Xbox 360, el rango cercano está disponible sólo en el sensor Kinect para Windows.

[9] "Microsoft Kinect SDK". [En línea]. Recuperado el 08 de marzo del 2013. Disponible en Web: <http://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges>
[10] "Depth Camera". [En línea]. Recuperado el 1 de marzo del 2013, Disponible en Web: <http://msdn.microsoft.com/en-us/library/hh438997.aspx>

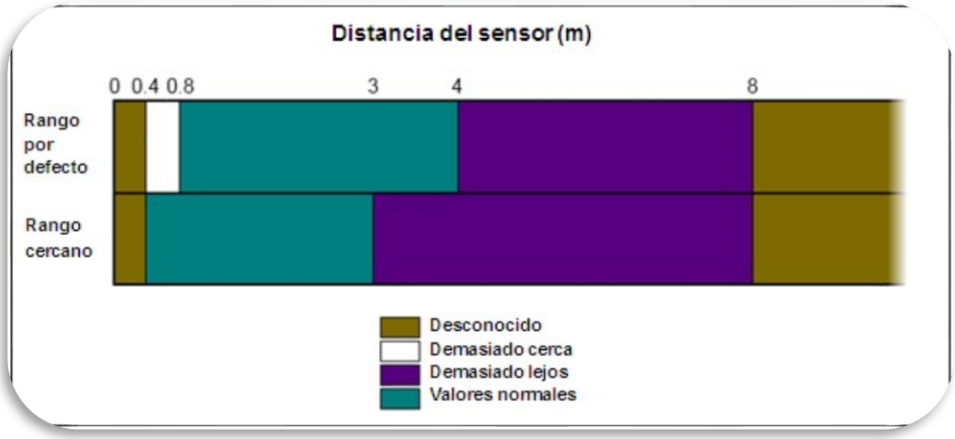


Figura 1.17: Rango de distancias admitidas por el sensor Kinect.[11]

1.4.4 Limitaciones[12]

El sensor Kinect tiene varias limitaciones que hacen que la profundidad de ciertas regiones de la escena no se pueda estimar o si se estima, la fiabilidad de los datos no es aceptable. Estas limitaciones vienen condicionadas tanto por factores internos, debidos a la arquitectura del dispositivo; como externos, debidos a la naturaleza de la escena.

Los puntos de luz no cubren de forma continua la superficie de los objetos (Figura 1.12), lo que conlleva a que algunos píxeles de la imagen de profundidad tienen que ser interpolados. Esto implica que el valor de profundidad de un píxel determinado tiene asociado un margen de error. Este margen es mayor cuanto más alejado está el objeto, puesto que, para una misma superficie, los puntos de luz están más separados.

A mayores distancias, los valores de profundidad devueltos para objetos cercanos entre sí tienden a ser muy similares. Sin embargo, si el objeto está a demasiada distancia del sensor, no se calcula ninguna distancia para ese punto (Figura 1.18).

Esto ocurre así, porque la potencia de luz del haz de infrarrojos se atenúa en el trayecto recorrido, haciendo que sea imperceptible para el sensor de infrarrojos.

La luz emitida por el proyector de infrarrojos, al impactar sobre un objeto, genera una sombra de éste a mayor distancia, como se puede apreciar en la Figura 1.19. El resultado es que no se puede determinar la profundidad en las zonas afectadas por dichas sombras. Esto se manifiesta como píxeles de valor cero (“zonas negras”) en la imagen de profundidad.

---

[11] “Depth Space Range”. [En línea]. Recuperado el 1 de marzo del 2013, Disponible en Web: <http://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges>

[12] CÓRDOVA, Fabricio. “Detección de robo/abandono de objetos en interiores utilizando cámaras de profundidad”. Universidad autónoma de Madrid, Diciembre 2012.

Figura 1.18: Efecto de distancias grandes en el Kinect[13]

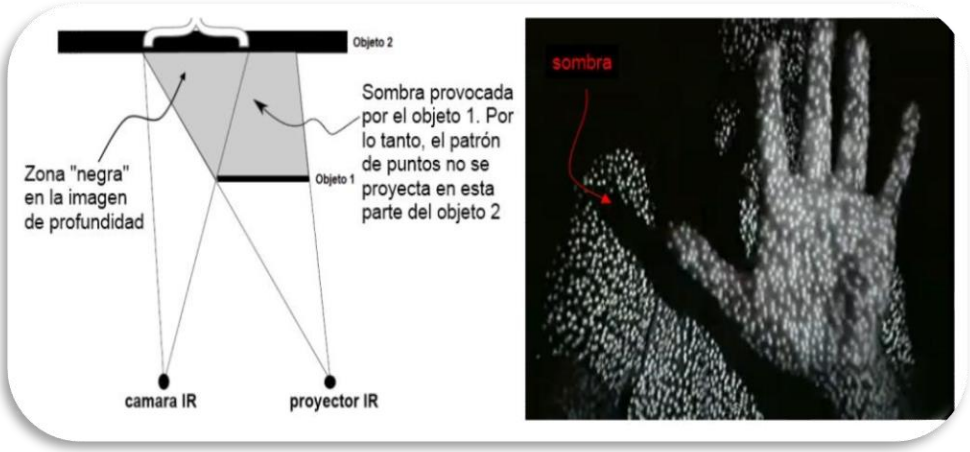


Figura 1.19: Efecto de las sombras en el Kinect[14]

Como ocurre con todo sistema óptico, la otra gran limitación viene determinada por las características de los objetos (factores externos). En general, atendiendo a la forma y propiedades de las superficies, se puede hacer la siguiente clasificación de objetos:

- Objetos translúcidos: los puntos de luz que impactan sobre éstos sufren una dispersión, haciendo indistinguible la deformación del patrón para el sensor.
- Objetos reflectantes**:** los puntos de luz tienden a impactar sobre el objeto reflejado.

Otra limitación radica cuando hay un cambio muy fuerte de luminosidad (luz solar) que impide que la luz infrarroja pueda ser detectada por el Sensor IR.

[13] "Limitaciones Kinect". [En línea]. Recuperado el 10 de marzo del 2013, Disponible en Web: <http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20121212FabricioACordovaLucero.pdf>
[14] "Detección de objetos con Kinect". [En línea]. Recuperado el 10 de marzo del 2013, Disponible en Web: <http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20121212FabricioACordovaLucero.pdf>

Así mismo, la inclinación de la superficie de los objetos respecto al proyector del haz de luz limita la detección de la profundidad. Si el rayo de luz es casi paralelo a la superficie no podrá incidir sobre la misma, haciendo imposible la estimación de la profundidad. En la Figura 1.20 se puede ver este efecto en la mesa de la esquina inferior derecha y en la superficie del suelo.

Figura 1.20: Efecto de superficie reflectante y paralela a los rayos de luz[15]

1.5 SOFTWARE PARA CONTROLAR EL SENSOR KINECT

Existen una gran variedad de librerías disponibles, cada uno con sus drivers propios para controlar el sensor Kinect. Esto significa que no es posible combinar dos librerías diferentes (de propietarios diferentes) para crear una aplicación concreta.

1.5.1 Microsoft Kinect SDK

Este es el kit de desarrollo oficial lanzado por la corporación de Microsoft. La documentación de Microsoft afirma que el rango de distancias válido para el sensor de profundidad es de 4 a 11.5 pies (1.2 a 3.5 metros). Los valores de profundidad para cada píxel son devueltos en milímetros en dos bytes (16 bits), pero sólo los 12 bits más significativos contienen la información "real" de profundidad. Microsoft no dice nada acerca de la precisión de estos valores (mayores de 3.5 metros); pero, como es lógico, las degradaciones en la información de profundidad serán mayores.

Algunas de las características son[16]:

- Soporte para audio.

[15] "Kinect". [En línea]. Recuperado el 10 de marzo del 2013, Disponible en Web: <http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20121212FabricioACordovaLucero.pdf>

[16] IRALDE L. Iñaki; PINA C. Alfredo. "Desarrollo de aplicaciones con Microsoft Kinect". Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación. Pamplona, Abril 2012.

- Soporta manejo del motor de inclinación.
- Tracking de todo el cuerpo.
- No necesita calibración previa.
- Funciona mejor con los Joints ocultos.
- Puede manejar varios sensores.
- Instalación sencilla.
- Incluye eventos que saltan cuanto un nuevo frame o trama está disponible.
- Licencia no comercial.
- No da soporte a otros sensores similares a Kinect.
- Sólo soporta Windows 7.

1.5.2 OpenNI SDK

Este SDK pertenece a PrimeSense, compañía líder en "interfaces naturales" y encargada de la fabricación del sensor de profundidad para el dispositivo Kinect. OpenNI es la organización que se encarga del desarrollo del software para todos los dispositivos fabricados por PrimeSense. Al igual que MSDK, el valor de profundidad se devuelve en dos Bytes y en milímetros[17].

1.5.3 Open Kinect

Antes de la aparición de los entornos de desarrollo oficiales (MSDK y OpenNI), desarrolladores de todo el mundo hicieron grandes esfuerzos de ingeniería inversa sobre la Kinect para poder interpretar y manipular los datos entregados por ésta. Una de las comunidades de desarrolladores que se ha consolidado fuertemente es precisamente OpenKinect, cuya labor está centrada en el desarrollo de libfreenect, el paquete software para controlar el sensor Kinect.

1.5.4 Microsoft Kinect SDK vs. Open Source

Entre las opciones de software de desarrollo de Kinect existen dos posibles librerías: librerías libres o librerías oficiales. Dentro de las primeras, la más conocida y la más utilizada es la llamada OpenNI que guarda compatibilidad con las librerías de OpenKinect[30]. En la Tabla 1.2 se presentan las ventajas del SDK de Microsoft y OpenNI y en la Tabla 1.3 las desventajas.

---

[17] "Kinect primeros pasos". [En línea]. Recuperado el 11 de marzo del 2013, Disponible en Web: <http://alumnos.elo.utfsm.cl/~flopezp/Kinect_v1.pdf> [30] "Microsoft SDK vs OpenNI", [En línea]. Recuperado el 09 de marzo del 2013. Disponible en Web: <http://www.argencon.org.ar/sites/default/files/123.pdf>

Tabla 1.2: Ventajas del SDK de Microsoft y del OpenNI.

SDK de Microsoft	SDK de OpenNI
Soporte para audio	Licencia incluye uso comercial.
Soporte para el motor de inclinación	Reconocimiento de gestos con las manos.
Seguimiento completo	Trackeo de cuerpo entero.
No necesita paso de calibración	Soporta Windows, Linux y MacOSX.
Soporta múltiples sensores	Calcula la rotación de las articulaciones
Instalación simple	Tiene eventos cuando un usuario ingresa o abandona el cuadro.

Tabla 1.3: Desventajas del SDK de Microsoft y del OpenNI.

SDK de Microsoft	**SDK de OpenNI**
Licencia para uso no comercial	Sin soporte para audio
Trackeo cuerpo enero (no exclusivo de manos)	Sin soporte para el motor de inclinación
Calcula posición de las articulaciones pero no rotación	Necesita de una fase de calibración
Solo para Windows 7	Intrincada instalación

ANEXO 2 (Base Kinect)

ANEXO 3

Modulo para Presentaciones de Alto Impacto.

ANEXO 4 (Blender)

Manual de Blender by Jose Guillen Granados

AMEXO 5 (Python)

Python tutorial release 3.2.3 by Guido Van Rossum
Python no muerde by Roberto Alsina
ANEXOS\Python\MANUAL.pdf
Pensando en Python II by by Diego Lz. De Ipiña Gz. De Artaza
Graficas en Python by Miguel Cárdenas Montes

ANEXO 6 (Programas)

ANEXO7 (Archivos de blender)

ANEXO 8 (Encuesta)

CON HERRAMIENTAS TECNOLOGICAS PARCIALES.

ENCUESTA AMBIENTE DE TRABAJO 1

NOMBRE:
CARGO:
INSTITUCION:

Marque con una "X" la respuesta que considere.

La información que se suministre en esta encuesta será usada con fines académicos, no se verá involucrado su nombre en el análisis de la misma. Esta información será relacionada al cargo descrito en la parte superior del mismo, es decir referenciaremos este como su espacio de trabajo (El "Cargo" no discrimina entre trabajadores o estudiantes).

1. **¿Es de utilidad la información suministrada en este módulo para su desempeño laboral?**

   SI____ NO____

2. **¿Aplica usted alguna metodología de desarrollo de presentaciones para difundir sus ideas de manera efectiva en su entorno laboral?**

   SI____ NO____ Uso mi propia metodología ____

   En caso de que su respuesta sea "NO", responda:

   ¿Por qué?:

   No conozco alguna metodología para el desarrollo de presentaciones efectivas ____

   Mi lugar de trabajo no necesita de esta clase de herramientas ____

   No encuentro útil este tipo de herramientas ____

   OTRO _____

3. **De 1 a 10 de su valoración sobre este módulo. Siendo 10 el máximo valor y 1 el mínimo.**

1	2	3	4	5	6	7	8	9	10

4. **¿Conoce usted alguna herramienta tecnológica para el desarrollo de sus presentaciones?**

   SI____ NO____ Uso mi propia herramienta ____

   En caso de que su respuesta sea "SI" o "Uso mi propia herramienta", responda:

   ¿Cual? _____

   ¿Es de su uso frecuente? _____