LEAP SIGN^{VR}: APLICACIÓN BASADA EN REALIDAD VIRTUAL PARA EL APOYO DE LA ENSEÑANZA DE LA LENGUA DE SEÑAS COLOMBIANA USANDO LEAP MOTION

JOHANNA CAROLINA SÁNCHEZ RAMÍREZ

CÓD. 2011103062

JUAN CAMILO SALAZAR GUEVARA

CÓD. 2011103058

UNIVERSIDAD PEDAGÓGICA NACIONAL

FACULTAD DE CIENCIA Y TECNOLOGÍA

DEPARTAMENTO DE TECNOLOGÍA

LICENCIATURA EN ELECTRÓNICA.

BOGOTÁ D.C. 2018

LEAP SIGN^{VR}: APLICACIÓN BASADA EN REALIDAD VIRTUAL PARA EL APOYO DE LA ENSEÑANZA DE LA LENGUA DE SEÑAS COLOMBIANA USANDO LEAP MOTION

JOHANNA CAROLINA SÁNCHEZ RAMÍREZ

JUAN CAMILO SALAZAR GUEVARA

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE LICENCIADOS EN ELECTRÓNICA

DIEGO MAURICIO RIVERA PINZÓN

DIRECTOR

UNIVERSIDAD PEDAGÓGICA NACIONAL

FACULTAD DE CIENCIA Y TECNOLOGÍA

DEPARTAMENTO DE TECNOLOGÍA

LICENCIATURA EN ELECTRÓNICA

BOGOTÁ D.C. 2018

Nota de aceptación.

_____

_____

_____

_____

_____

_____

_____

Firma del Director del trabajo de Grado.

_____

Firma del jurado.

_____

Firma del jurado.

Bogotá D.C, 2018.

1. Información General	
Tipo de documento	Trabajo de grado.
Acceso al documento	Universidad Pedagógica Nacional. Biblioteca Central
Título del documento	LEAP SIGN^{VR}: Aplicación basada en realidad virtual para el apoyo de la enseñanza de la lengua de señas colombiana usando Leap Motion.
Autor(es)	Sánchez Ramírez, Johanna Carolina; Salazar Guevara, Juan Camilo
Director	Rivera Pinzón, Diego Mauricio
Publicación	Bogotá. Universidad Pedagógica Nacional, 2018. 91 p.
Unidad Patrocinante	Universidad Pedagógica Nacional
Palabras Claves	DACTILOLOGÍA; LENGUA DE SEÑAS; LEAP MOTION; MATLAB; UNITY

2. Descripción

Leap SignVR es una aplicación diseñada para el apoyo de la enseñanza de la lengua de señas colombiana con el objetivo de que los usuarios puedan interactuar, aprender y fortalecer esta lengua con el uso del Leap Motion y a través de un conjunto de software (“Orion”, MatLab y Unity), todo esto desarrollado en un entorno de Realidad Virtual.

La idea principal es que el usuario retroalimente las señas realizadas con ayuda de las gafas de realidad virtual y con el sensor Leap Motion, éste último se encarga de detectar y representar las manos en un ambiente de realidad virtual, y se encuentra ubicado en el visor de las gafas, para permitir que el usuario manipule la aplicación.

Con respecto al diseño, en esta aplicación se trabajó con la dactilología de la lengua de señas colombiana, usando el Leap Motion y MatLab para capturar la posición de cada seña, los datos adquiridos son un conjunto de sistemas de coordenadas espaciales, los cuales fueron almacenados en matrices con el fin de enseñarle a una red neuronal a identificar el alfabeto de la lengua de señas. Por otro lado, se realizó un código que permitió comunicar a MatLab con Unity, para transmitirle los datos aprendidos por la red neuronal y así poder establecer un funcionamiento en los escenarios elaborados en Unity, los cuales fueron diseñados para ilustrar y evaluar la lengua de señas colombiana, además de retroalimentar al usuario a medida que recorre los escenarios.

3. Fuentes

Aggarwal, R., Swetha, S., Namboodiri, A. M., Sivaswamy, J., & Jawahar, C. V. (2015). Online handwriting recognition using depth sensors. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, (págs. 1061 - 1065). Tunis.

Bairathi , D., & Gopalani, D. (2017). Opposition-Based Sine Cosine Algorithm (OSCA) for Training Feed-Forward Neural Networks. *13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, (págs. 438-444). Jaipur.

Bernal Villamarin, S. C., Cantor Morales, D. A., Ávila Reyes, C. A., & Sánchez, C. A. (2016). Application design sign language colombian for mobile devices VLSCApp (Voice Colombian sign language app) 1.0. *2016 Technologies Applied to Electronics Teaching (TAEE)*, (págs. 1-5). Sevilla.

Chuan, C. H., Regina, E., & Guardino, C. (2014). American Sign Language Recognition Using Leap Motion Sensor. *13th International Conference on Machine Learning and Applications*, (págs. 541 - 544). Detroit.

Demircioğlu, B., Bülbül, G., & Köse, H. (2016). Turkish Sign Language recognition with Leap Motion. *Signal Processing and Communication Application Conference (SIU)*, (págs. 589 -592). Zonguldak.

Ebrahim Al-Ahdal, M., & Nooritawati, M. T. (2012). Review in Sign Language Recognition Systems. *Symposium on Computers & Informatics (ISCI)*, (págs. 52 - 57). Penang.

Freeman, I., Salmon, J., & Coburn, J. (2016). CAD Integration in Virtual Reality Design Reviews for Improved Engineering Model Interaction. *ASME 2016 International Mechanical Engineering Congress and Exposition*, (pág. V011T15A006). Phoenix, Arizona.

Garcia, M. G., Luis , C. I., & Samonte, M. J. (2016). E-tutor for Filipino Sign Language. *2016 11th International Conference on Computer Science & Education (ICCSE)*, (págs. 223-227). Nagoya.

Geetha, M., Manjusha, C., Unnikrishnan, P., & Harikrishnan, R. (s.f.). A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images. *2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications, IEEE-C2SPCA 2013*, (pág. 2013).

Herrera, V. (2005). Adquisición Temprana de Lenguaje de Signos y Dactilología. *Revista Psicopedagógica*, 2-10.

Inkscape. (2018). https://inkscape.org/es/ .

INSOR. (2006). *Diccionario básico de la lengua de señas colombiana.* Bogotá: Instituto Nacional para Sordos e Instituto Caro y Cuervo.

Jeisson Pérez. (2015). Sings2Me. *WiseWare S.A.S*.

Khalil, M. A., & Kotaiah, B. (2017). Implementation of agile methodology based on SCRUM tool. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, (págs. 2351-2357). Chennai.

Ladrón de Guevara, M. Á. (2018). *Técnicas de comunicación con personas dependientes en instituciones.* Logroño: Tutor Formación.

Leap Motion. (2018). *https://www.leapmotion.com/*.

Ling, H., & Rui, L. (2016). VR glasses and leap motion trends in education. *2016 11th International Conference on Computer Science & Education (ICCSE)*, (págs. 917-920). Nagoya.

Liou, W.-K., & Chang, C.-Y. (2018). Virtual reality classroom applied to science education. *23rd International Scientific-Professional Conference on Information Technology (IT)*, (págs. 1 - 4). Zabljak.

MathWorks. (2018). https://www.mathworks.com/products/matlab.html.

Medjram, S., Babahenini, M. C., Taleb-Ahmed, A., & Bed Ali, Y. M. (2016). *Real-time wrist localization in color images based on corner analysis.* New York.

Modanwal, G., & Sarawadekar, K. (2018). *A RobustWrist Point Detection Algorithm using Geometric Features.* Varanasi.

Mohandes, M., Aliyu, A. S., & Deriche, M. (2014). Arabic sign language recognition using the leap motion controller. *23rd International Symposium on Industrial Electronics (ISIE)*, (págs. 960 - 965). Istanbul.

Motion, L. (2018). https://www.leapmotion.com/.

Oculus. (2018). https://www.oculus.com/.

Orza, J. G. (2002). Neuropsicología cognitiva de la lengua de signos: una piedra de toque para el estudio del lenguaje, la visión, las emociones y el movimiento. *: Revista de psicología general y aplicada: Revista de la Federación Española de Asociaciones de Psicología*, 89-104.

Parra, C. (2010). Educación inclusiva : Un modelo de educación para todos. *Revista Isees*, 73-84.

Paulraj, M. P., Yaacob, S., bin Zanar Azalan, M. S., & Palaniappan, R. (2010). A phoneme based sign language recognition system using skin color segmentation. *2010 6th International Colloquium on Signal Processing & its Applications*, (págs. 1 - 5). Mallaca City.

Pertusa Venteo, E., & Fernandez Viader, M. D. (2005). *El valor de la mirada: sordera y educación.* Barcelona.

Ravikiran, J., Mahesh, K., Mahishi, S., Dheeraj, R., Sudheender, S., & Nitin, V. (2009). Finger Detection for Sign Language Recognition. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I. IMECS 2009.* Hong Kong.

Ren, Z., Yuan, J., & Zhang, Z. (2014). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *2011 Proceedings of the 19th ACM international conference on Multimedia - MM '11*.

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, (págs. 864-869). Greater Noida.

Steinicke, F. (2016). *Being Really Virtual Immersive Natives and the Future of Virtual Reality.* Cham: Springer.

SubVRsive. (2018). *https://subvrsive.com/announcing-project-asl-american-sign-language-mixed-reality/.*

Tablada, C. J., & Torres, G. A. (2015). Redes Neuronales Artificiales. *Revista de Educación Matemática*, 22-30.

TechSmith. (2018). https://www.techsmith.com/video-editor.html.

Thepade, S. D., Kulkarni, G., Narkhede, A., Kelvekar, P., & Tathe, S. (2013). Sign language recognition using color means of gradient slope magnitude edge images. *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, 216 - 220.

Unity Technologies. (2018). *https://unity3d.com/es*.

Vince, J. (2011). *Introduction to Virtual Reality.* Tunbridge Wells: Springer.

Wibowo, M. D., Nurtanio, I., & Ahmad Ilham, A. (2017). Indonesian sign language recognition using leap motion controller. *11th International Conference on Information & Communication Technology and System (ICTS)*, (págs. 67 -72). Surabaya.

Xu, Y., Gu, J., & Tao, Z. (2009). Bare Hand Gesture Recognition with a Single Color Camera. *2nd International Congress on Image and Signal Processing*, (págs. 1 - 4). Tianjin.

Yeh, W. Y., Tseng, T. H., Hsieh, J. W., & Tsai, C. M. (2016). Sign language recognition system via Kinect: Number and english alphabet. *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, (págs. 660 - 665). Jeju.

4. Contenidos

LEAP SIGN^{VR} se diseñó con base al objetivo general “Diseñar una aplicación educativa para el apoyo a la enseñanza básica de la lengua de señas colombiana basada en realidad virtual”, del cual se desglosaron los objetivos específicos “Crear espacios de formación virtual, que sirvan como apoyo a la enseñanza e interpretación de la lengua de señas colombiana”, “Seleccionar movimientos cotidianos y básicos de la lengua de señas colombiana, para que el usuario los reitere usando únicamente las manos y dedos”, “Diseñar cuatro actividades donde el usuario pueda practicar y seguir su proceso de aprendizaje de la lengua de señas” y “Diseñar un algoritmo que permita al usuario retroalimentarse de cada actividad de apoyo culminada”.

Los antecedentes consultados fueron la base de LEAP SIGN^{VR}, los cuales van desde aplicaciones en dispositivos móviles, traductores de la lengua de señas usando el Leap Motion y la dactilología de la lengua de señas americana.

El marco teórico refleja conceptos usados para el desarrollo de la aplicación, tales como: Entorno Virtual, MatLab, Unity, Leap Motion, Lengua de Señas, Orion.

LEAP SIGN^{VR} se desarrolló partiendo del uso del sensor Leap Motion en un ambiente de Realidad Virtual, enfocándose en el apoyo a la enseñanza del alfabeto de la lengua de señas colombiana, para ello se representó las señas en Orion y Matlab para contrastar similitudes entre las señas reales y las simuladas, luego se procedió a realizar una red neuronal Feed Forward para entrenar los datos de las posiciones cartesianas de cada seña con el fin de reconocer que seña está realizando el usuario en tiempo real.

Por último, se realizaron los escenarios en un motor de videojuegos llamado Unity, los cuales permiten la interacción, la observación y evaluación del alfabeto de la lengua de señas colombiana en tiempo real.

5. Metodología

La metodología implementada en este proyecto fue *SCRUM.*

Scrum comprende un desarrollo ágil, en el cual se realizan trabajos seccionados, es decir que parten de una idea macro y se va desglosando en partes más pequeñas para realizar el trabajo por ciclos, estos ciclos son conocidos como Sprints. Cada Sprint maneja un tema específico que se puede desarrollar por días o por semanas, dependiendo del trabajo que se va a realizar, además cada Sprint se somete a una iteración que consiste en una revisión minuciosa del trabajo realizado, esta revisión constante permite avanzar a otro Spring, siempre y cuando la meta se haya alcanzado.

En este proyecto se realizaron tres Sprints de trabajo, los cuales fueron:

- Primer Sprint: Contraste del alfabeto de LSC con las señas simuladas en “Orion”.
- Segundo Sprint: Representación gráfica y captura de datos en MatLab.
- Tercer Sprint: Visualización, captura y validación de la LSC en MatLab.

Estos Sprints sirvieron para realizar un trabajo organizado y flexible.

6. Conclusiones

A continuación, se exponen una serie de resultados que muestran tanto el proceso como las dificultades que se presentaron durante el desarrollo del proyecto. A modo de recomendaciones se formularán pautas para la continuación de este.

Se diseñó un salón de clase porque brinda un entorno pedagógico similar a un salón real.

Se seleccionó el alfabeto de la lengua de señas colombiana, porque el conjunto de consonantes y vocales le permite a la comunidad sorda poder comunicarse a través del deletreo, además como es una aplicación educativa brinda la posibilidad de que las personas oyentes aprendan esta lengua de signos.

El diseño de las cuatro actividades: exploración de los escenarios, observación de los vídeos, completar las palabras y la evaluación de las señas, fueron resultado de una serie de investigaciones, donde se observaron diferentes herramientas que se enfocan en el aprendizaje unidireccional, es decir, una comunicación entre el humano-máquina por medio de imágenes y videos que no permite una retroalimentación por parte del usuario, por este motivo se desarrollaron actividades en Realidad Virtual que permiten un aprendizaje bidireccional, es decir, el usuario puede interactuar, aprender, reforzar y retroalimentar el alfabeto de la lengua de señas colombiana en tiempo real.

El desarrollo de los algoritmos en el proyecto es un punto que resaltar, porque con ello se logró la retroalimentación y evaluación del alfabeto de la lengua de señas colombiana, a través de la captura de las posiciones cartesianas espaciales de la mano derecha, e identificando cada seña por medio del algoritmo de la resta de la posición de la palma con cada uno de los dedos, con el fin de que la seña no se distorsione de acuerdo con la posición en el espacio.

Después de realizar las evaluaciones a los tipos de entrenamientos Levenberg-Marquardt con un acierto del 68.8% y Regularización Bayesiana con un acierto de 83.7%, al momento de realizar el alfabeto de la lengua de señas colombiana, se concluyó que el mejor tipo de entrenamiento para este caso es la Regularización Bayesiana.

El sensor Leap Motion está limitado a la hora de detectar la posición de los dedos en la Realidad Virtual, debido a que el sensor está ubicado sobre las gafas oculus y el jugador debe realizar las señas a sí mismo, a causa de esto las señas del alfabeto de la lengua de señas colombiana: M, N y Ñ no se pueden realizar satisfactoriamente, puesto que el sensor no las diferencia, por la posición de los dedos y el dorso de la mano, además era incomodo realizar estas señas de frente al sensor Leap Motion por consiguiente se recomienda hacer una pequeña modificación en la rotación a las señas mencionadas.

LEAP SIGN^{VR} es una aplicación que está enfocada en la parte educativa para apoyar la enseñanza del alfabeto de lengua de señas colombiana, con el fin de reducir las brechas de la comunicación entre la comunidad sorda y la oyente, y por parte de la tecnología LEAP SIGN^{VR} está enfocada en las experiencias que genera el entorno de la Realidad Virtual beneficiando el uso y la interacción con los usuarios.

Elaborado por:	Sánchez Ramírez, Johanna Carolina; Salazar Guevara, Juan Camilo
Revisado por:	Rivera Pinzón, Diego Mauricio

Fecha de elaboración del Resumen:	27	08	2018

Contenido

Lista de Figuras

Lista de Tablas

RESUMEN:

Leap Sign^{VR} es una aplicación diseñada para el apoyo de la enseñanza de la lengua de señas colombiana (LSC) con el objetivo de que los usuarios puedan interactuar, aprender y fortalecer esta lengua con el uso del Leap Motion y a través de un conjunto de software ("Orion", MatLab y Unity), todo esto desarrollado en un entorno de Realidad Virtual (RV).

La idea principal es que el usuario retroalimente las señas realizadas con ayuda de las gafas de realidad virtual y con el sensor Leap Motion, éste último se encarga de detectar y representar las manos en un ambiente de realidad virtual, y se encuentra ubicado en el visor de las gafas, para permitir que el usuario manipule la aplicación.

Con respecto al diseño, en esta aplicación se trabajó con la dactilología[1] de la LSC, usando el Leap Motion y MatLab para capturar la posición de cada seña, los datos adquiridos son un conjunto de sistemas de coordenadas espaciales, los cuales fueron almacenados en matrices con el fin de enseñarle a una red neuronal a identificar el alfabeto de la lengua de señas. Por otro lado, se realizó un código que permitió comunicar a MatLab con Unity, para transmitirle los datos aprendidos por la red neuronal y así poder establecer un funcionamiento en los escenarios elaborados en Unity, los cuales fueron diseñados para ilustrar y evaluar la LSC, además de retroalimentar al usuario a medida que recorre los escenarios.

Finalmente, para trabajar con la aplicación el usuario deberá contar con las siguientes especificaciones mínimas: un computador con tarjeta gráfica NVDIA GTX 960, una CPU Intel i5, un sistema operativo Windows 8, una RAM 8GB, una salida de video

[1] Representación manual de las letras del alfabeto, que sirve para la comunicación entre personas.

HDMI 1.3, puertos USB 3.0 y dos USB 2.0, además un Leap Motion y unas gafas de realidad virtual (Oculus Rift), como se ve en la Figura 1, la aplicación retroalimentará las señas efectuadas por el usuario al mismo tiempo que se realizan, el objetivo es ayudar a las personas a comprender esta lengua.

Es importante resaltar que, a diferencia de otras aplicaciones, ésta no traduce ni interpreta la lengua de señas.

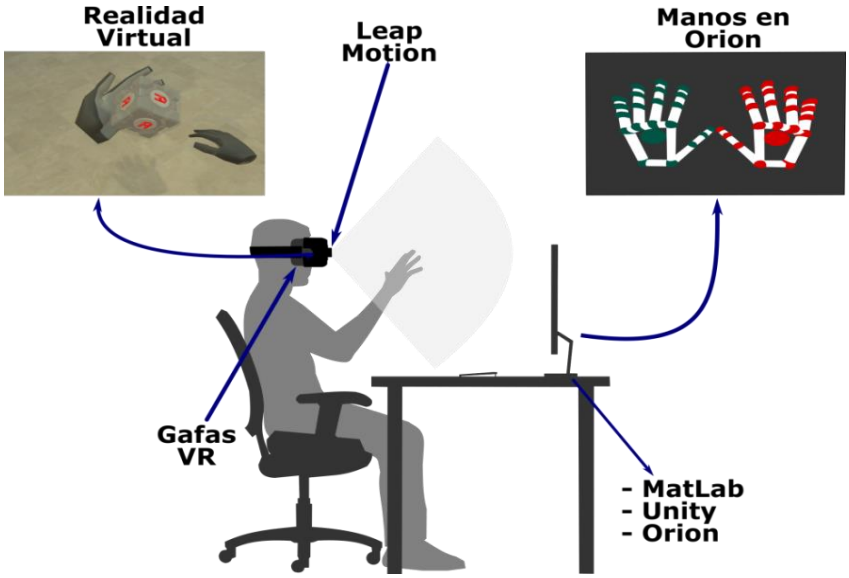


Figura 1. Elementos con los que se interactúa en la aplicación.
Fuente: Adaptado de www.leapmotion.com

Capítulo I. Introducción

La tecnología es una fuente que impulsa el desarrollo y avance de una sociedad, no hay duda de que la tecnología avanza día a día, y más con la aparición de nuevas tecnologías como la Realidad Virtual y el sensor Leap Motion que promueven la innovación en el ámbito educativo actual, con métodos de enseñanza práctica *(Vince, 2011)*.

La realidad virtual se ha extendido rápidamente en varios campos como en la educación. Una aplicación con fines educativos facilita la forma de aprendizaje en este caso de la lengua de señas, la cual fue desarrollada en un ambiente virtual, con la ayuda de software y hardware como: Oculus Rift y el sensor Leap Motion, los cuales permiten que los estudiantes puedan practicar esta lengua de señas para poder acercarse a la comunidad con capacidades diferentes, no oyentes *(Steinicke, 2016)*.

El uso del Leap Motion y "Orion" han servido para desarrollar varias aplicaciones que favorecen la interacción social, un ejemplo es la elaboración de traductores de señas en

diferentes partes del mundo, que han ayudado a personas con capacidades diferentes a ingresar tanto al mundo laboral como educativo y a favorecer la comunicación entre personas sordas con las personas oyentes *(Wibowo, Nurtanio, & Ahmad Ilham, 2017)*, *(Mohandes, Aliyu, & Deriche, 2014)*, *(Demircioğlu, Bülbül, & Köse, 2016)*, *(Chuan, Regina, & Guardino, 2014)*.

1.1 Planteamiento del Problema.

En Colombia existen aplicaciones enfocadas para que el usuario aprenda la lengua de señas a través de imágenes o videos, estas aplicaciones fueron creadas con el fin de suplir la necesidad de comunicarse con personas de capacidades diferentes, además sirve como medio para el aprendizaje de una nueva lengua, un ejemplo de estas aplicaciones es Sings2me[2]; esta aplicación sirve para comunicarse con otras personas por medio de un chat de señas, la interfaz gráfica de la aplicación es un teclado que contiene las señas del alfabeto, otro ejemplo es Centro de Relevo[3]; un servicio de comunicación para personas sordas, diseñado para la inclusión social de ésta comunidad a través de una plataforma que contiene videos educativos, un diccionario y además ofrecen un servicio de video llamada que facilita a las personas con capacidades diferentes establecer comunicación con instituciones por medio de intérpretes. Pero estas aplicaciones no le brindan al usuario la posibilidad de saber si los gestos que realizan son los correctos, por este motivo, se diseñó una aplicación para el apoyo de la enseñanza de la LSC, que le permita a las personas

[2] Aplicación realizada en LSC. http://www.enter.co/chips-bits/apps-software/esta-app-te-ensena-lenguaje-de-senas/
[3] Aplicación de MIN TICS Colombia. http://discapacidadcolombia.com/index.php/centro-de-relevo

experimentar, interactuar y reforzar sobre el alfabeto de la lengua de señas en un ambiente de RV.

1.2 Justificación.

La educación es un elemento primordial de la sociedad, ya que compone la base del aprendizaje, la construcción del conocimiento personal, así mismo es un derecho fundamental para todo ser humano, por lo que la educación ha evolucionado para ser inclusiva, es decir, la educación le abre puertas a las personas que tienen discapacidades diferentes para generar una igualdad de oportunidades en la participación física y social (Parra, 2010).

De la misma forma, el desarrollo de la tecnología virtual abre múltiples posibilidades en el ámbito educativo, ya que es una herramienta que genera experiencias por ejemplo, con el surgimiento de las gafas de Realidad Virtual y el sensor Leap Motion trabajando en conjunto forman un apoyo al aprendizaje, se ha buscado trabajar con estas dos herramientas tecnológicas para entender las cosas, refiriéndose a que los estudiantes experimenten, interactúen y aprendan con juegos a través de una realidad inmersiva. (Ling & Rui, 2016).

Por tal razón, se realizó una aplicación que busca generar una inclusión en la educación a las personas sordas y oyentes a través de la tecnología (Gafas de Realidad Virtual y el sensor Leap Motion) y el apoyo de la enseñanza del alfabeto de la LSC inmerso en un ambiente virtual, que brinda la posibilidad de aprender o reforzar la lengua de señas colombiana mediante la observación, la práctica y la evaluación.

1.3 Delimitación del Problema

Leap Sign^{VR} está dirigida para una población sorda y oyente, que cuenten con motricidad fina en las manos. Por ser una aplicación en Realidad Virtual se recomienda usar un espacio de trabajo de tres metros cuadrados sin obstáculos para que el usuario tenga la mejor experiencia en el desplazamiento dentro del entorno virtual, es muy importante resaltar que esta aplicación va dirigida a personas mayores de 13 años[4] que no sufran de astigmatismo[5].

1.4 Objetivos

1.4.1 Objetivo General.

- Diseñar una aplicación educativa para el apoyo a la enseñanza básica de la lengua de señas colombiana basada en realidad virtual.

1.4.2 Objetivos Específicos.

- Crear espacios de formación virtual, que sirvan como apoyo a la enseñanza e interpretación de la lengua de señas colombiana.

- Seleccionar movimientos cotidianos y básicos de la lengua de señas colombiana, para que el usuario los reitere usando únicamente las manos y dedos.

[4] https://allvreducation.blogspot.com/2016/01/10-recomendaciones-previas-en-torno-al.html
[5] https://hipertextual.com/2016/03/realidad-virtual-miopia-hipermetropia-astigmatismo

- Diseñar cuatro actividades donde el usuario pueda practicar y seguir su proceso de aprendizaje de la lengua de señas.

- Diseñar un algoritmo que permita al usuario retroalimentarse de cada actividad de apoyo culminada.

1.5 Antecedentes

En los últimos tiempos la ciencia y la tecnología se han enfocado en crear dispositivos, aplicaciones y artefactos para generar una inclusión social para aquellas personas que se encuentran con alguna capacidad diferente, en este caso problemas auditivos, la lengua de señas también ha sido objeto de trabajo para las nuevas tecnologías. Los trabajos relacionados a continuación son antecedentes con fines académicos cercanos al trabajo propuesto.

1.5.1 Aplicaciones educativas usando la lengua de señas.

Se han desarrollado aplicaciones para dispositivos móviles en los cuales se pretende enseñar la lengua de señas de forma visual (Imágenes y videos), donde el usuario puede interactuar a través de escenarios, un ejemplo es el proyecto basado en LSC que consiste en una aplicación que tiene un diccionario con palabras básicas de las señas colombianas, además consta de una entrada de voz o texto, para ser llevado a una salida que cuenta con un modelado en 3D de un avatar que plasma las señas de las palabras que se ingresaron *(Bernal Villamarin, Cantor Morales, Ávila Reyes, & Sánchez, 2016)*. Otro trabajo relacionado a este tipo de aplicaciones consiste en un diccionario de 50 señas, que ilustra

por medio de imágenes y videos la realización de cada seña, además cuenta con una serie de escenarios de prueba para evaluar el trabajo realizado por el usuario *(Garcia, Luis , & Samonte, 2016)*.

1.5.2 Lengua de señas ayudado de imágenes.

La captura y análisis de imágenes fue una alternativa para realizar el reconocimiento de las señas a través del color, ciertos autores se basaron en el contorno de la mano, usando métodos como: la primera derivada de la función Gaussiana para analizar las imágenes a través de pixeles y texturas, otro método usado fue el Operador Gradiente que pone una máscara en la figura o seña realizada para ser analizada por vectores, también se usa para extraer el borde de la imagen, es decir, el contorno de la mano *(Aggarwal, Swetha, Namboodiri, Sivaswamy, & Jawahar, 2015)*; *(Ravikiran, y otros, 2009); (Thepade, Kulkarni, Narkhede, Kelvekar, & Tathe, 2013)*. Otra técnica usada fue por medio de la segmentación de los colores RGB, que determinan por medio de un método binario que silueta o seña se está realizando, 0 para un fondo sin detección de imágenes y 1 para indicar que hay una figura *(Paulraj, Yaacob, bin Zanar Azalan, & Palaniappan, 2010)* . Un último ejemplo de estos trabajos es el basado en el análisis de la posición, rotación y el ángulo de la muñeca de la mano, obteniendo los puntos del contorno de ésta *(Medjram, Babahenini, Taleb-Ahmed, & Bed Ali, 2016)*; *(Modanwal & Sarawadekar, 2018)*.

1.5.3 Lengua de señas ayudado del sensor Kinect.

Con la llegada del sensor de profundidad Kinect surgieron nuevas oportunidades de interacción entre el computador y el ser humano. El Kinect permitió trabajar con imágenes de color e imágenes de profundidad simultáneamente.

En la representación de la lengua de señas usando el Kinect se encontraron varios trabajos, entre estos se encuentra la construcción de un sistema que reconoce el contorno de la mano, mediante el uso de una manilla negra ubicada en la muñeca, con el fin de que el sensor detecte de forma segura la posición de la mano *(Ren, Yuan, & Zhang, 2014)*. Otra estrategia implementada fue extraer datos globales y locales de la mano, los datos globales fueron tomados usando el método *Axis of Least Inertia* y los datos locales a través de las características de la mano, posición de las yemas de los dedos, la palma y la muñeca, con el objetivo de representar la lengua de señas india *(Geetha, Manjusha, Unnikrishnan, & Harikrishnan)*. Un último proyecto se centró en la captura de datos de la posición, el color y la profundidad para detectar el área de la mano y luego hacer la identificación de los números y el alfabeto en la lengua de señas realizado en tiempo real *(Yeh, Tseng, Hsieh, & Tsai, 2016)*.

1.5.4 Aplicación educativa sobre la lengua de señas americana (LSA).

La empresa XR Labs realizó un trabajo basado en el LSA usando Unity y Leap Motion, la intención de este proyecto fue crear una interacción entre la máquina y el usuario con el fin de hacer una retroalimentación de la lengua de señas que realiza el

usuario. Esta aplicación le muestra al usuario tarjetas que tienen imágenes del alfabeto y palabras básicas del LSA, que éste debe realizar para luego ser evaluado[6] *(SubVRsive, 2018)*.

1.5.5 Traductores de la lengua de señas ayudado del Leap Motion.

Existen varios dispositivos y aplicaciones que traducen la lengua de señas a nivel mundial a continuación, se reúnen varios documentos que tienen en común la elaboración de traductores bidireccionales de la lengua de señas en tiempo real ayudado del sensor Leap Motion, estos trabajos fueron desarrollados en países como: Estados Unidos, Indonesia, Arabia Saudita y Turquía. Estos han trabajado con la lengua de señas propia, es decir, cada país y cada región tienen su propia lengua de señas.

Para la elaboración de cada uno de los traductores de la lengua de señas se utilizó el Leap Motion, el método de trabajo fue realizar la identificación de la posición de las manos de cada seña, los datos obtenidos de las identificaciones fueron implementadas en un sistema de redes neuronales, usando como métodos de entrenamiento *Back Propagation* y *Bayesiano* consiguiendo como resultado un rango de efectividad del 89% al 99% para las señas sin movimiento. Algunas de las aplicaciones realizadas fueron hechas en Java, otros trabajos fueron realizados en "Orion" *(Wibowo, Nurtanio, & Ahmad Ilham, 2017); (Mohandes, Aliyu, & Deriche, 2014)*; *(Demircioğlu, Bülbül, & Köse, 2016)*; *(Chuan, Regina, & Guardino, 2014)*.

[6] El desarrollo y resultado de esta aplicación no tiene una difusión académica que permita al investigador acceder a su contenido, ya que es una iniciativa de carácter privado, sin embargo, se puede consultar en el siguiente enlace: https://subvrsive.com/announcing-project-asl-american-sign-language-mixed-reality/

Capítulo II. Marco Teórico y Metodología

2.1 Marco Teórico

Leap Sign^{VR} es el producto de una serie de elementos que aportaron para el desarrollo visual, el funcionamiento y la experiencia virtual, con el objetivo de que el usuario retroalimente su experiencia sobre el alfabeto de LSC, los temas que se muestran a continuación son elementos que se implementaron en el desarrollo de la aplicación.

2.1.1 Realidad Virtual (RV).

La RV es la capacidad de sumergir al usuario en un mundo virtual en 3D y permitirle interactuar con modelos digitales *(Freeman, Salmon, & Coburn, 2016)*, otros autores definen la RV como la unión del entorno real con un entorno virtual que producen nuevos ambientes y visualizaciones donde los elementos físicos y digitales conviven e interactúan en tiempo real *(Liou & Chang, 2018)*.

2.1.2 Lengua de Señas.

Es un lenguaje gestual que permite la comunicación entre las personas, (Comúnmente usada en las comunidades sordas y mudas), transmite de forma visual una serie de gestos o patrones que generan una idea o un mensaje, los gestos son considerados como los movimientos específicos de las manos, cara y cuerpo *(Ebrahim Al-Ahdal & Nooritawati, 2012)*.

Las señas se pueden clasificar dependiendo de las posiciones de las manos; están las posturas: figuras estáticas que se dan debido a la ubicación de la mano sin movimiento y los gestos: una serie de posturas realizadas sucesivamente en un determinado tiempo *(Xu, Gu, & Tao, 2009)*.

2.1.3 Gafas Oculus Rift.

Es un dispositivo que tiene un aspecto de visor, con dos pantallas OLED, que permiten una resolución Full HD, lleva incorporado unos auriculares, que envuelve al usuario en un ambiente virtual a través del sonido, como se presenta en la Figura 2, además consta de dos sensores que permiten rastrear la posición del usuario, logrando tener un desplazamiento completo de 6 grados de libertad en un espacio determinado de Realidad Virtual *(Oculus, 2018)*.

Figura 2. Gafas de Realidad Virtual Oculus Rift.
Fuente: Tomado de https://commons.wikimedia.org/wiki/File:Oculus-Rift-CV1-Headset-Front_with_transparent_background.png

2.1.4 Sensor Leap Motion.

Es un dispositivo que tiene tres leds infrarrojos capaces de detectar objetos tridimensionales a través de un espectro de luz, éstos están separados de una forma adecuada para obtener una iluminación constante dependiendo de la luz en el ambiente, además consta de dos sensores ópticos que son los encargados de capturar las imágenes, pueden llegar a una velocidad de trabajo hasta de 200 cuadros por segundo dependiendo del equipo con el que se use, estos sensores son de tipo monocromático y sensibles a la luz infrarroja, el Leap Motion también tiene un microcontrolador , el cual se encarga de guardar los datos obtenidos por los sensores ópticos, para ser leídos por el software "Orion", tiene una ranura USB para ser conectado a un computador *(Wibowo, Nurtanio, & Ahmad Ilham, 2017).* Se presenta en la Figura 3.

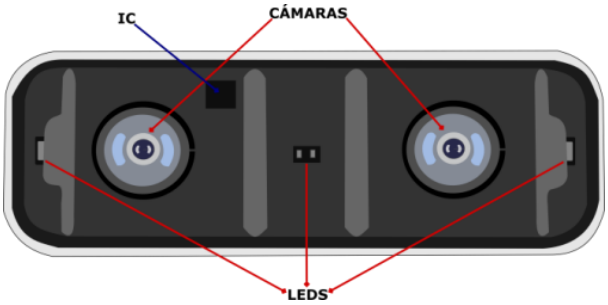


Figura 3. Partes internas del Leap Motion.
Fuente: Adaptado de www.leapmotion.com

La forma física del Leap Motion cuando esta desconectado es de pantalla oscura, al momento de ser conectado al computador se enciende un LED verde que indica que se puede usar el sensor, además se iluminan los leds infrarrojos, Figura 4. Este dispositivo es compatible con las gafas de realidad virtual (Oculus, HTC). En la página oficial de Leap Motion[7] se encuentran librerías, módulos, documentos y ejemplos que pueden trabajar en diferentes lenguajes de programación (C++, C#, Python) y plataformas de trabajo (Unity, Unreal, Java).

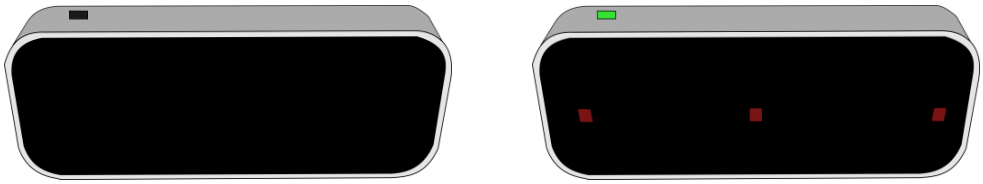


Figura 4. Leap Motion Apagado (izq), encendido (der).
Fuente: Elaboración Propia.

[7] www.leapmotion.com

2.1.5 SDK “Orion”.

Es un software que permite detectar y representar las manos de una forma virtual como se muestra en la Figura 5, y tiene la capacidad de dar la ubicación de las coordenadas de un sistema cartesiano espacial, de cada dedo y de la palma de la mano.

Para el funcionamiento del software, se descarga el *Orion Beta versión 4.0.0+52173* el cual contiene una carpeta llamada *LeapDevelopmentkit*[8] que se encuentra en la página oficial de Leap Motion. La versión usada para este trabajo fue *Unity Core Assets 4.4.0*, además en la página se encuentran los módulos de trabajo: *Interactive Engine*, *Hands Module* y *Graphic Renderer*, los cuales se pueden descargar por separado *(Leap Motion, 2018)*.

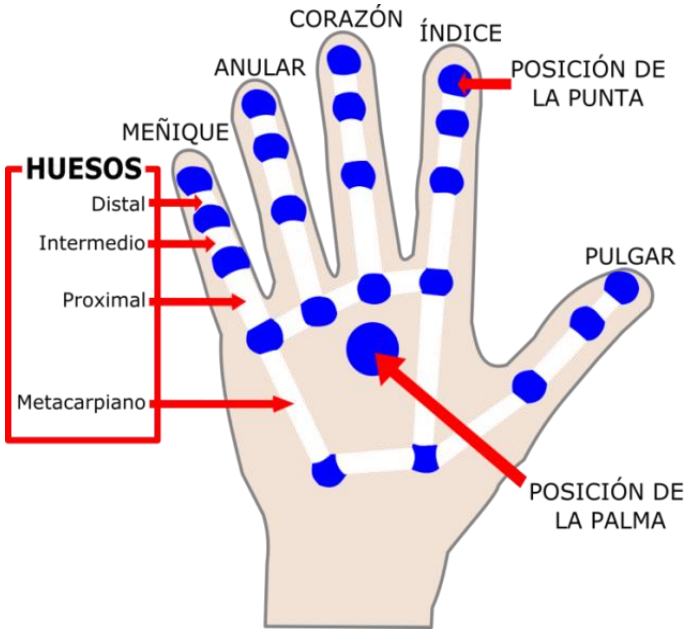


Figura 5. Representación gráfica de la mano derecha en Unity con ayuda del SDK “Orion”.
Fuente: Adaptado de www.leapmotion.com

[8] www.developer.leapmotion.com

2.1.6 MatLab.

Es un software que permite realizar análisis matemáticos a través de herramientas o toolboxes, el usado en este trabajo fue el *Neural Network*, el cual realiza la clasificación, la regresión, la agrupación, la reducción de dimensionalidad, el pronóstico de series de tiempo, el modelado y control de sistemas dinámicos. La versión usada en la aplicación fue el Matlab R2015b, esto debido a que los desarrolladores de Leap Motion crearon un archivo compatible para esta versión *(MathWorks, 2018)*.

2.1.7 Redes Neuronales Artificiales (RNA).

Las RNA son un diseño semejante a las redes neuronales del ser humano, las cuales tienen un grupo de neuronas biológicas conectadas entre sí, de igual forma las RNA tienen la misma estructura y funcionamiento.

Una RNA es un método para solucionar problemas de forma individual o grupal por medio de otras funciones, además tiene la habilidad de capturar datos para organizarlos, clasificarlos o predecir sus respuestas, tienen como finalidad minimizar los errores y reajustarlos de forma dinámica mientras la RNA se adapta, los cálculos se ajustan en los pesos que se encuentran en las interconexiones entre las neuronas *(Tablada & Torres, 2015)*.

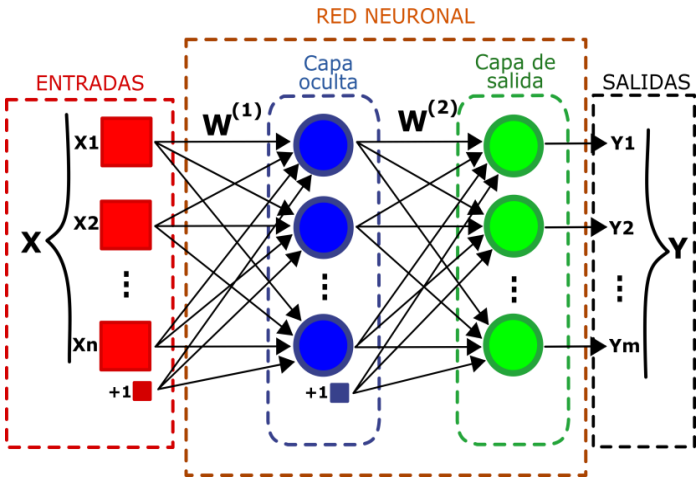


Figura 6. Estructura básica de una Red Neuronal Artificial.
Fuente: Adaptado de www.sites.google.com/site/mayinteligenciartificial/unidad-4-redes-neuronales

Existen varios tipos de RNA como: Perceptron, Feed Forward, Radial Basis, Recurrent, Deep Feed Forward, Auto Encoder. En este documento se hará énfasis en Feed Forward, ya que es la red neuronal que se usó en la aplicación.

2.1.8 Red Neuronal Feed Forward FANN.

Es una de las RNA más sencillas y usadas en la ciencia y la tecnología, resuelve problemas no lineales, la forma estructural de un FANN es constituida por tres capas, una capa de entrada, una capa oculta y una capa de salida, como se ve en la Figura 7, cada capa tiene una cantidad *n* de neuronas, las cuales van conectadas entre las capas.

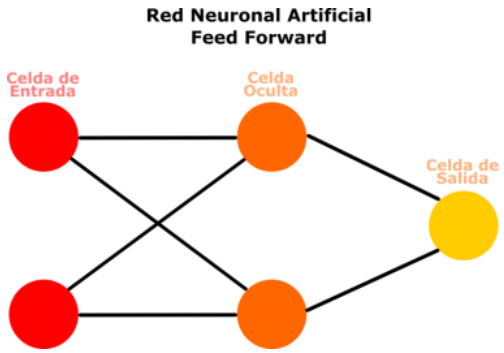


Figura 7. Estructura de una RNA Feed Forward. Adaptado de http://www.microsiervos.com/archivo/ordenadores/poster-redes-neuronales.html

Una FNNA cuando es entrenada es capaz de aprender patrones o instrucciones, debido a que realiza cambios en los pesos cada vez que se realiza un entrenamiento *(Bairathi & Gopalani, 2017)*.

2.1.9 Entrenamiento Regularización Bayesiana.

Es un algoritmo de entrenamiento del toolbox de MatLab conocido como trainbr[9], que se encarga de actualizar los valores de los *pesos* y *bias* de la red neuronal, para luego combinarlos y así obtener un resultado de entrenamiento de la red, además se busca obtener un valor de error cuadrático medio cercano a cero, y que la regresión en la salida de la red sea igual a uno *(MathWorks, 2018)*.

2.1.10 Unity.

Es un motor de desarrollo para el diseño de video juegos multiplataforma, es usado en este proyecto para el trabajo de modelos en 3D en un entorno de Realidad Virtual,

[9] Entrenamiento regularización bayesiana.

además trabaja en conjunto con el sensor Leap Motion y las gafas de Realidad Virtual Oculus Rift, por último, la versión usada para este proyecto fue el Unity 2017.3.1f1 *(Unity Technologies, 2018)*.

2.1.11 Monodevelop.

Es un editor de códigos de Unity, en el cual se puede realizar códigos con diferentes clases de lenguajes de programación como C# y Java *(Unity Technologies, 2018)*.

2.1.12 Inkscape.

Es un editor profesional de gráficos vectoriales, tiene una interfaz gráfica sencilla y amigable para el usuario, este programa está disponible para Windows, Mac OS X y Linux. Es libre y de código abierto. Este programa tiene como objetivo proporcionar herramientas donde se puede crear y editar gráficas, logotipos e ilustraciones entre otros[10] *(Inkscape, 2018)*.

2.1.13 Camtasia.

Es un editor de videos muy conocido gracias a que cuenta con una interfaz gráfica sencilla de usar en la edición de videos. Camtasia también es usado en la creación de tutoriales, grabaciones y presentaciones, todo esto es posible ya que tiene herramientas como: *Recorder* que permite grabar la pantalla en segundo plano y *Studio editor* que es la

[10] https://inkscape.org/es/

herramienta multimedia que permite al usuario agregar y ordenar videos, audios, textos y animaciones[11] *(TechSmith, 2018)*.

2.2 Metodología

En el campo empresarial y tecnológico se ha diseñado una metodología que ayuda a desarrollar proyectos con mayor efectividad, flexibilidad, con menor tiempo y que además enriquece con saberes al grupo de trabajo. Esta metodología es conocida como *SCRUM*.

SCRUM comprende un desarrollo ágil, en el cual se realizan trabajos seccionados, es decir que parten de una idea macro y se va desglosando en partes más pequeñas para realizar el trabajo por ciclos, estos ciclos son conocidos como Sprints. Cada Sprint maneja un tema específico que se puede desarrollar por días o por semanas, dependiendo del trabajo que se va a realizar, además cada Sprint se somete a una iteración que consiste en una revisión minuciosa del trabajo realizado, esta revisión constante permite avanzar a otro Spring, siempre y cuando la meta se haya alcanzado *(Srivastava, Bhardwaj, & Saraswat, 2017)*.

SCRUM se integra por:

- Un manager o un Scrum Master, es un miembro del equipo que tiene el papel de guiar y facilitar la ejecución del producto.
- Scrum team, encargados de realizar las tareas de cada Sprint.

[11] https://www.techsmith.com/video-editor.html

- Los tiempos de entrega de un Sprint son acordados entre el Scrum Team y el Scrum Master, estos tiempos pueden ser días o hasta semanas *(Khalil & Kotaiah, 2017)*.

En la Figura 8, se muestra el desarrollo de una metodología SCRUM básica.

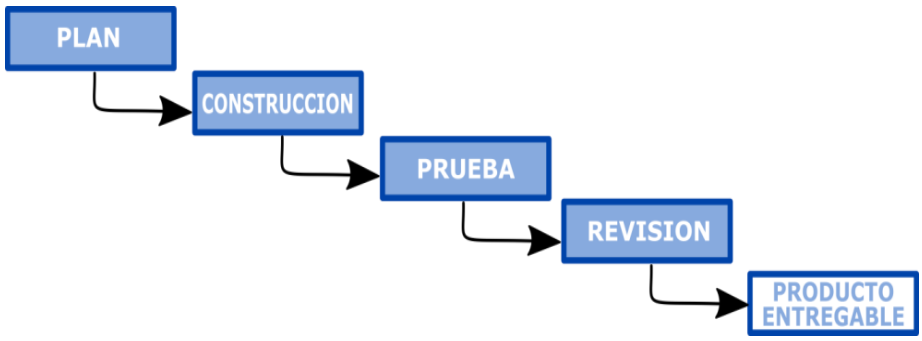


Figura 8. Modelo básico de un SCRUM.
Adaptado de: (Srivastava, Bhardwaj, & Saraswat, 2017).

2.2.1 Metodología SCRUM aplicada en el proyecto.

ROLES

- Scrum Master: El director del trabajo de grado.
- Scrum Team: Constituido por dos estudiantes.

COMPONENTES

- Pila Producto: Lista de requisitos del proyecto, para este trabajo se necesitó de: "Orion", Leap Motion, Gafas Oculus, MatLab, y Unity.
- Pila Sprint: Fracciona el trabajo en tareas más pequeñas.
  - Primer Sprint: Contraste del alfabeto de LSC con las señas simuladas en "Orion".

- Segundo Sprint: Representación gráfica y captura de datos en MatLab.
- Tercer Sprint: Visualización, captura y validación de la LSC en MatLab.

REUNIONES:

- Planificación de Sprint:

El equipo planifica los requisitos y prioridades para la elaboración de la pila Sprint.

- Reunión semanal: (Duración 40 minutos)

Es dirigida por el Scrum Master donde se responden: preguntas

¿Qué se hizo la semana pasada?

¿Cuál es el trabajo de esta semana?

¿Qué dudas hay?

- Retrospectiva de Sprint:

  Es moderada por el Scrum Master, se elabora un análisis del Sprint y se presenta el siguiente Sprint.

Por último, la metodología SCRUM se ve reflejada en la Figura 9 que brinda una idea general del trabajo realizado.

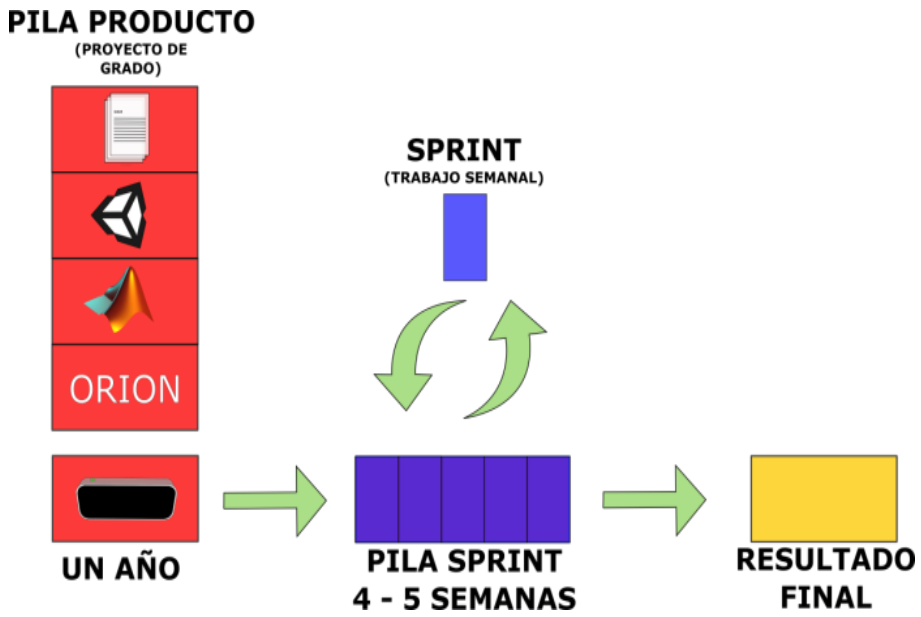


Figura 9. Metodología SCRUM aplicada al proyecto.
Fuente: Adaptado de https://es.slideshare.net/FlowersInSpace/introduccion-a-scrum-con-caso-prctico-1516220

Capítulo III. Desarrollo

Leap Sign^{VR} fue desarrollado con el objetivo de apoyar el aprendizaje del alfabeto de la lengua de señas colombiana en Realidad Virtual, para ello se tomó como referencia el diccionario básico de la lengua de señas colombiana que presenta el Ministerio de Educación y el Instituto Nacional para Sordos (INSOR)[12].

Para generar ese resultado se procedió a realizar unos sprints que marcarán el avance del trabajo, en el primer sprint se seleccionaron las señas con las cuales se implementó este proyecto: el alfabeto de la lengua de señas colombiana, éste al mismo tiempo fue representado en el software "Orion", donde se elaboró una comparación detallada entre el alfabeto de la lengua de señas colombiana físicas con las señas simuladas con el fin de determinar la fidelidad del software a la hora de representar las señas, el segundo sprint fue la captura y representación de las señas en MatLab, con este trabajo se logró observar las posiciones de la mano derecha a través de un sistema cartesiano espacial, y además se extrajeron las posiciones de cada seña para implementar una red neuronal con

[12] http://www.insor.gov.co/descargar/diccionario_basico_completo.pdf

el propósito de identificar en tiempo real la seña realizada y en el tercer sprint se elaboró la comunicación entre MatLab y Unity para el control de datos, además del diseño de los escenarios en Realidad Virtual, los mecanismos de trabajo, la programación necesaria para el funcionamiento y las características principales de la aplicación.

3.1 Sprint I: Contraste del alfabeto de LSC con las señas simuladas en “Orion”.

En este sprint se realizó una revisión de la LSC las cuales fueron consultadas en el diccionario básico de la lengua de señas presentado por INSOR, donde se seleccionó el alfabeto manual o dactilología que se muestra en la figura 10. Este alfabeto es la base inicial de la comunicación de las personas sordas antes de iniciar un aprendizaje formal, asimismo la dactilología sirve para deletrear nombres de personas y lugares para los que no cuentan con una seña determinada, también se usa para aclarar una seña en particular (Pertusa Venteo & Fernandez Viader, 2005); (Ladrón de Guevara, 2018).

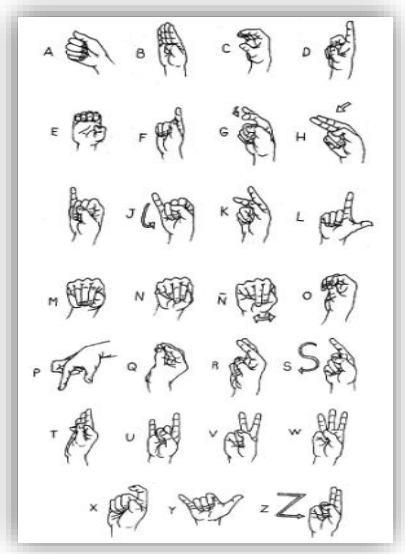


Figura 10. Alfabeto Manual
Fuente: Tomado de INSOR.

Por otro lado, para la enseñanza de la lengua de señas es importante la expresión corporal (Figura 11), ya que la posición y altura de las articulaciones pueden generar diferentes significados, para el caso del alfabeto se debe doblar el codo quedando la mano a la altura de la barbilla sin expresiones faciales (INSOR, 2006).

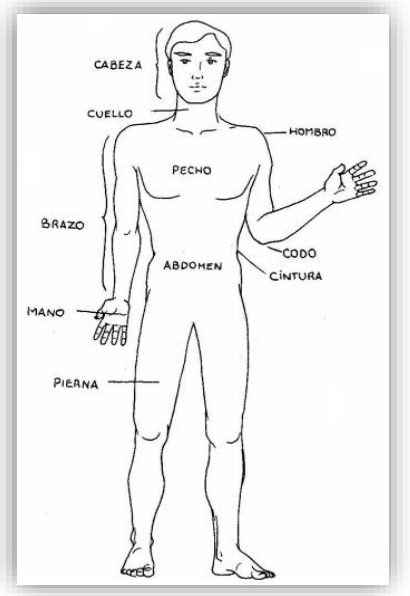


Figura 11. Punto de referencia del cuerpo para la articulación de la seña.
Fuente: Tomado de INSOR.

En el desarrollo de este proyecto fue necesario definir dos estrategias de trabajo: primero la expresión corporal para realizar las señas y segundo la ubicación del sensor.

La primera estrategia fue definir la expresión corporal a la hora de realizar el alfabeto en LSC, para ello fue necesario hacer un giro 180° a cada una de las señas del alfabeto consultadas en el diccionario de INSOR, figura 12, para que el usuario ubique la mano al frente de la cara, ya que la aplicación es desarrollada en un entorno virtual y el usuario se encuentra en primera persona.

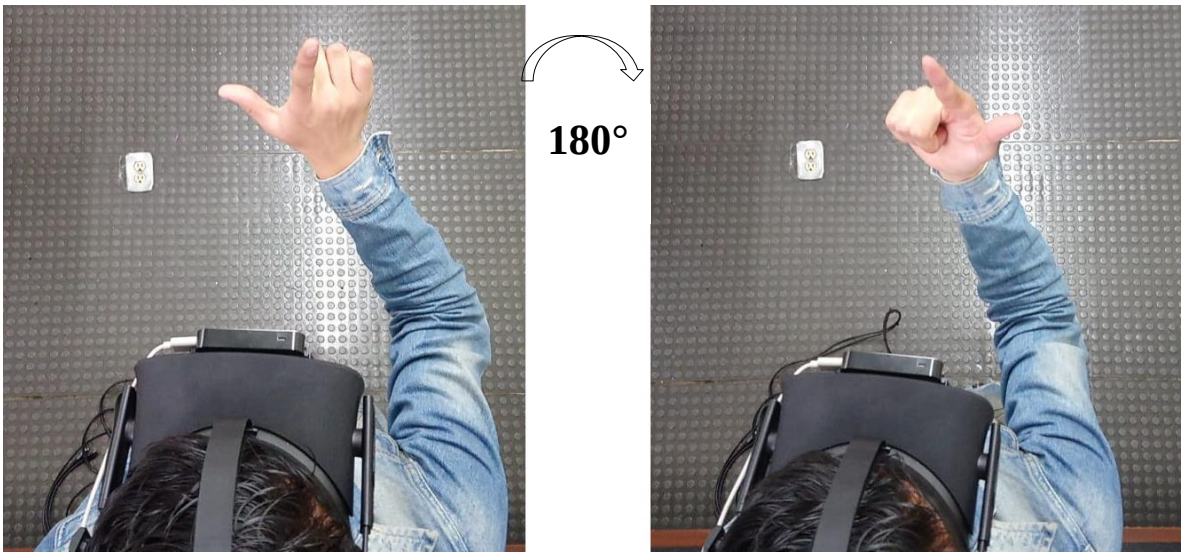


Figura 12. Posición del usuario para realizar las señas, izq. Ejecución real de la seña, der. Estrategia realizada en el proyecto Fuente: Adaptado de INSOR.

La segunda estrategia para apoyar la enseñanza del alfabeto de LSC fue ubicar el sensor Leap Motion en el visor de las gafas Oculus Rift con la intensión de que el usuario pueda visualizar las manos en la realidad virtual sin importar en el lugar donde se encuentre, esto significa que el usuario va a visualizar las manos cada vez que las ponga al frente de la cara y también cuando realice una actividad relacionada con el alfabeto de la LSC.

Dicho la anterior, las señas fueron representadas a través del software “Orion” con ayuda del sensor Leap Motion, de allí se procedió a contrastar entre las señas realizadas físicamente con las representadas en “Orion” con el fin de determinar la fiabilidad de las señas en el programa, el criterio de comparación se basó en la asociación visual entre las imágenes, tomando como referencia la ubicación de los dedos y de la palma de la mano en cada seña realizada, como resultado se obtuvo que varias de las señas simuladas se asemejaron a las señas reales, tales como B, C, U, entre otras. Por otro lado, la simulación

de las señas M, N y Ñ que se asemejaron en el software “Orion” a causa de la posición del dorso de la mano y la disposición similar de los dedos en la LSC, por ejemplo, el gesto que representa la letra M es con la mano cerrada y con los dedos índice, medio y anular estirados hacia abajo, a diferencia de la N que solo tiene los dedos índice y medio estirados hacia abajo, y la Ñ se diferencia por el movimiento lateral que se debe hacer, como se presenta en la Figura 13, por tanto, se optó por realizar una rotación de 90° para la seña M, una rotación de 45° para la seña N y dejar la seña Ñ sin movimiento, como se presenta en la Figura 14, con este cambio, se obtuvo una mejor visualización del dorso de la mano y de la ubicación de los dedos en cada una de las señas en el software “Orion”.

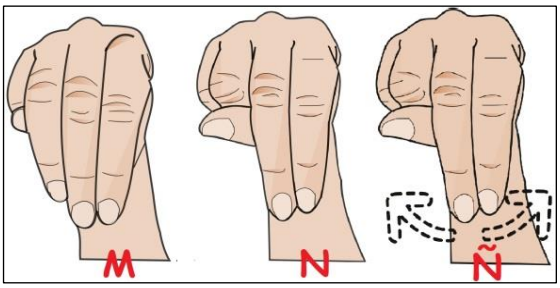


Figura 13. Señas M, N y Ñ de la LSC original.
Fuente: Elaboración propia.

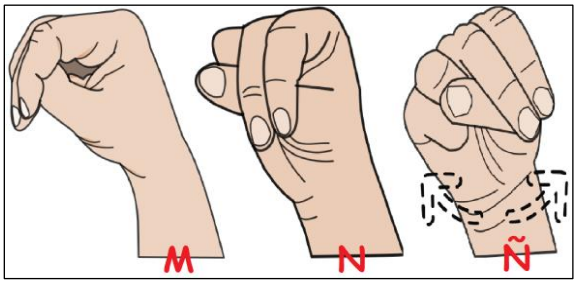


Figura 14. Señas M, N y Ñ con una rotación.
Fuente: Elaboración propia.

Por otro lado, la postura de la seña P como presenta más movimientos y dificultad al realizarla ocasiona que el sensor Leap Motion no tenga visualización total de los dedos, dado que los dedos pulgar e índice deben ir estirados hacia abajo, y en seguida la yema del dedo medio se debe colocar encima del hueso proximal del dedo índice, formando un arco y ocultando los dedos anular y meñique, como se muestra en la Figura 15.

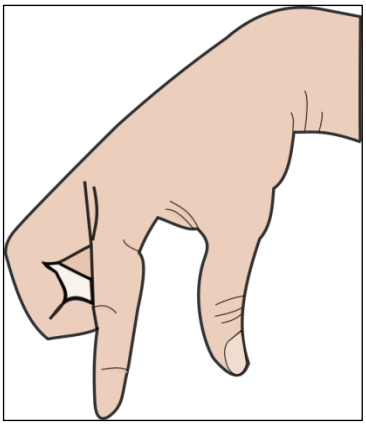
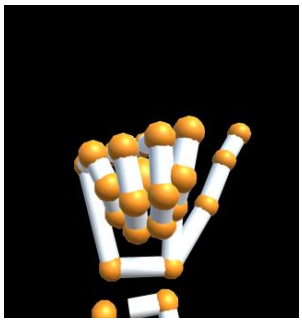
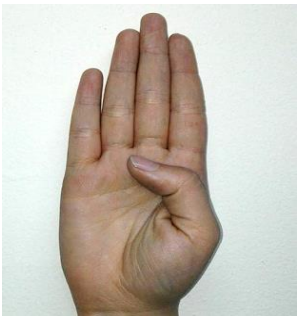
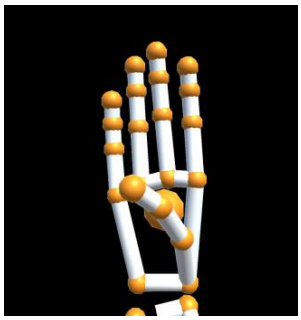
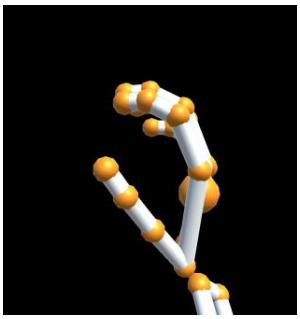
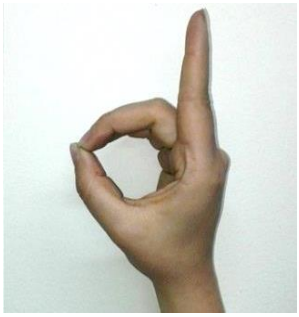
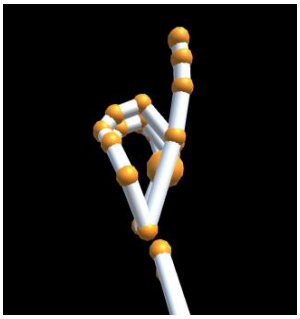
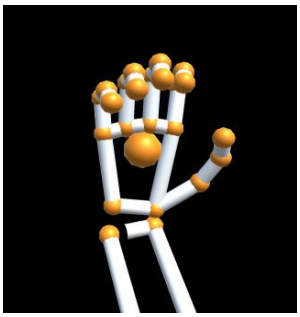
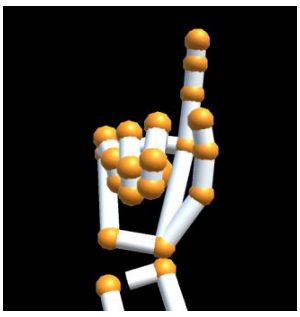
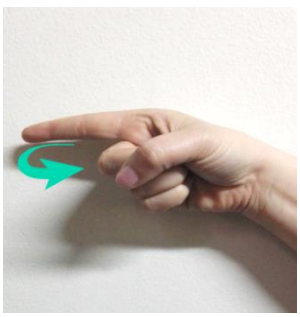
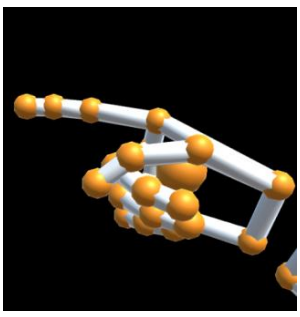


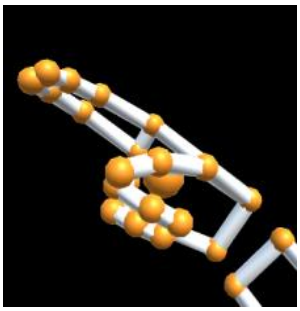
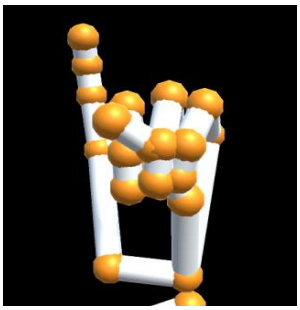
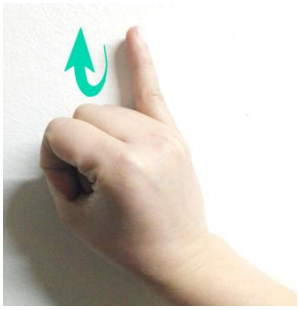
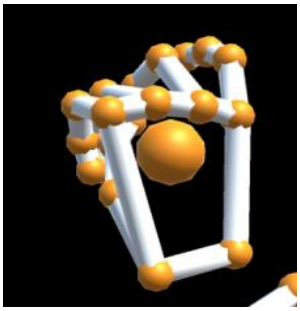
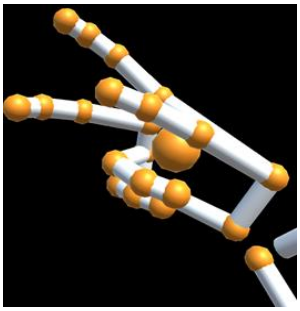
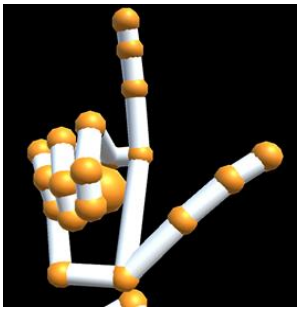
Figura 15. La seña P de la LSC, donde los dedos Anular y Meñique son cubiertos por el dedo Medio generando que el Leap Motion no detecte la posición de éstos.
Fuente: Elaboración propia.

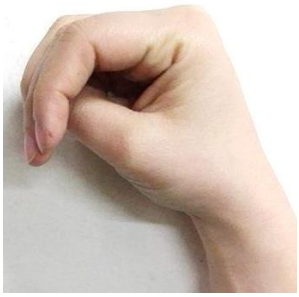
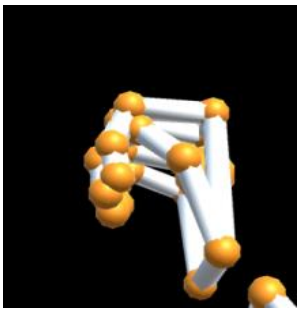
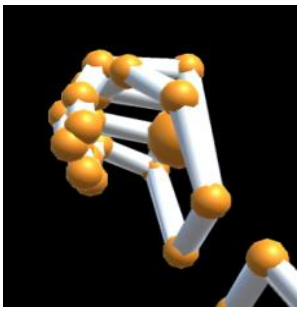
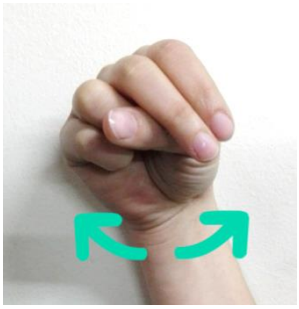
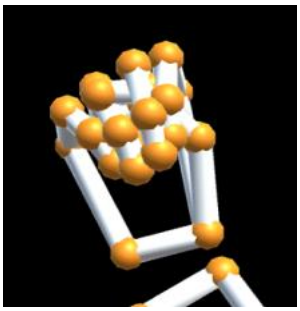
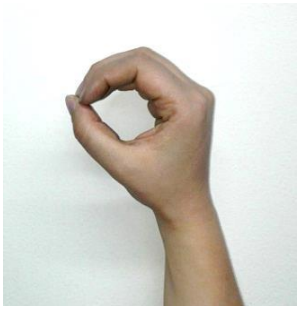
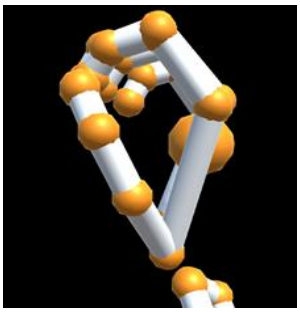
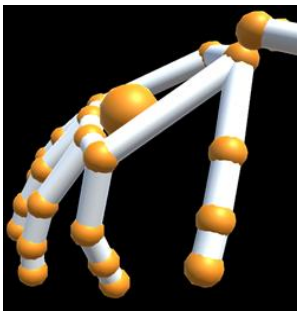
Ahora se realiza un contraste de las 27 letras del alfabeto de la LSC real con la representación virtual de éstas en Unity por medio de “Orion”, presentadas en la Tabla 1:

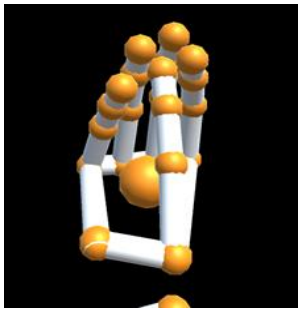
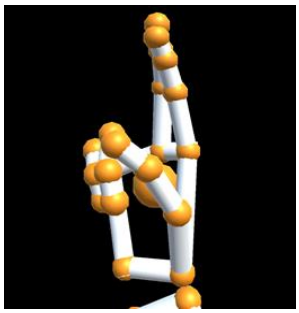
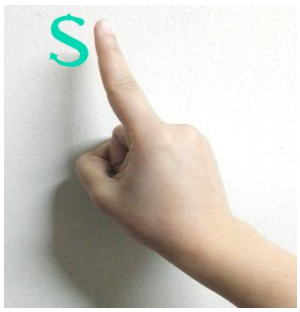
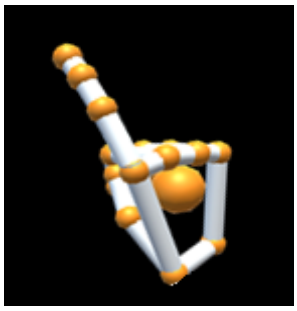
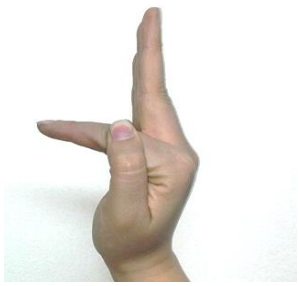
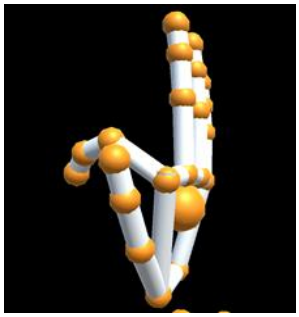
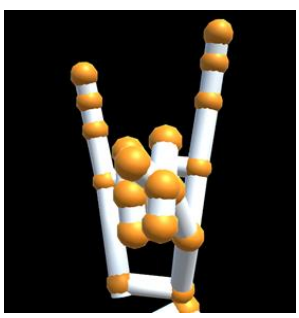
Tabla 1. Contraste del Alfabeto de la LSC real con las señas realizadas en "Orion".

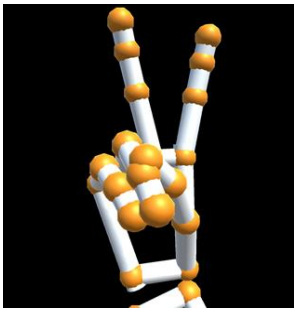
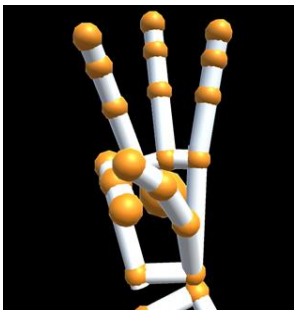
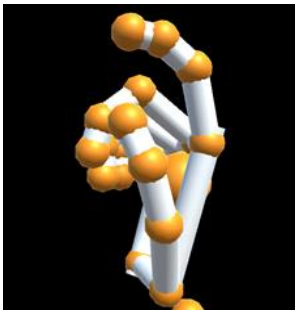
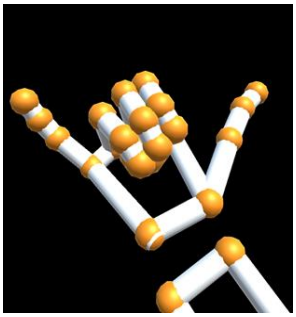
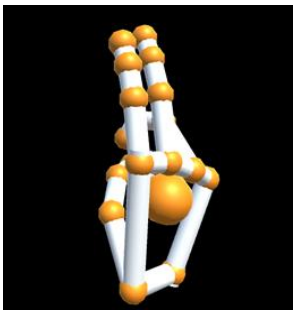
LETRA	SEÑA REAL (a)	SEÑA EN "ORION" (b)	CONTRASTE
A			La representación de la seña A en “Orion” es parecida a la seña real (a).
B			La representación de la seña B en “Orion” es parecida a la seña real (a).

C			La representación de la seña C en “Orion” es semejante a la seña real (a), aunque la posición del dedo pulgar en la imagen (b) no genera la apertura presentada en la imagen (a).
D			La representación de la seña D en “Orion” es parecida a la seña real (a), pero las falanges del dedo pulgar en la imagen (b) no generan la curvatura de la imagen (a).
E			La representación de la seña E en “Orion” se asemeja a la seña real (a), pero la falange del dedo pulgar en la imagen (b) no se dobla como en la imagen (a).
F			La representación de la seña F en “Orion” se aproxima a la seña real (a), pero la posición del dedo pulgar en la imagen (b) no es la ideal, ya que se encuentra encima del dedo índice.
G			La representación de la seña G en “Orion” es semejante a la seña real (a).

H			La representación de la seña H en “Orion” es semejante a la seña real (a).
I			La representación de la seña I en “Orion” es parecida a la seña real (a).
J			La representación de la seña J en “Orion” no se asemeja a la seña real, debido a que el dorso de la mano cubre los dedos, generando una representación de la mano cerrada por parte del software al momento de detectar la seña.
K			La representación de la seña K en “Orion” es parecida a la seña real (a), pero el dedo medio de la imagen (b) no se inclina como el de la imagen (a).
L			La representación de la seña L en “Orion” es fiel a la seña real.

M			La representación de la seña M en “Orion” no se parece a la seña real (a), porque el sensor Leap Motion no detecta las posiciones de los dedos cuando la mano se encuentra cerrada y los dedos juntos. Además, se asemeja a la seña N y Ñ.
N			La representación de la seña N en “Orion” no se parece a la seña real (a), porque el sensor Leap Motion no detecta las posiciones de los dedos cuando la mano se encuentra cerrada y los dedos juntos. Además, se asemeja a la seña M y Ñ.
Ñ			La representación de la seña Ñ en “Orion” no se parece a la seña real (a), porque el sensor Leap Motion no detecta las posiciones de los dedos cuando la mano se encuentra cerrada y los dedos juntos. Además, se asemeja a la seña M y Ñ.
O			La representación de la seña O en “Orion” se aproxima a la seña real (a), pero las falanges del dedo pulgar en la imagen (b) no generan la curvatura de la imagen (a).
P			La representación de la seña P en “Orion” no se asemeja a la real (a), debido a que el dorso de la mano cubre los dedos, generando una representación errónea por parte del software al momento de detectar la mano.

Q			La representación de la seña Q en “Orion” es parecida a la seña real (a).
R			La representación de la seña R en “Orion” es parecida a la seña real (a).
S			La representación de la seña S en “Orion” es parecida a la seña real (a).
T			La representación de la seña T en “Orion” se asemeja a la seña real (a), aunque la falange del dedo índice en la imagen (b) no se estira completamente como el dedo de la imagen (a).
U			La representación de la seña U en “Orion” es semejante a la seña real (a).

V			La representación de la seña V en “Orion” es semejante a la seña real (a).
W			La representación de la seña W en “Orion” es semejante a la seña real (a).
X			La representación de la seña X en “Orion” se parece a la seña real (a), aunque las falanges del dedo índice en la imagen (b) no se logran doblar como el dedo de la imagen (a).
Y			La representación de la seña Y en “Orion” es fiel a la seña real.
Z			La representación de la seña Z en “Orion” es semejante a la seña real (a).

3.2 Sprint II: Visualización, captura y validación de la LSC en MatLab.

En este sprint se trabajó tres partes importantes del proyecto, la primera parte se refiere a la visualización de la mano en MatLab ayudado del sensor Leap Motion, con el fin de determinar las variaciones existentes con las señas del alfabeto, la segunda parte fue la captura de las posiciones del sistema cartesiano espacial de cada seña en una matriz, la tercera y última parte fue la validación de las señas a través del uso de las Redes Neuronales de MatLab, las cuales ayudaron a la identificación del alfabeto de la LSC.

3.2.1 Visualización de la LSC en MatLab.

Para establecer comunicación entre Leap Motion y MatLab se remitió a una documentación encontrada en *GibHub*[13], una plataforma de desarrollo colaborativo que pone en funcionamiento el sensor Leap Motion con MatLab, por medio de una carpeta llamada *matleap-master* que contiene archivos que extraen información del Leap Motion, datos como la posición, velocidad, dirección e identificación de las manos.

[13] Se elaboró la interfaz en MatLab para trabajar con el Leap Motion. https://github.com/jeffsp/matleap

De los datos extraídos, se trabajó con la posición de las yemas de la mano derecha, cabe resaltar que la información proporcionada por los archivos desarrollados por *GibHub* son numéricos, en cambio, el código que se realizó para este proyecto consiste en plasmar los datos numéricos de la posición en un sistema cartesiano espacial, como se muestra en la Figura 16.

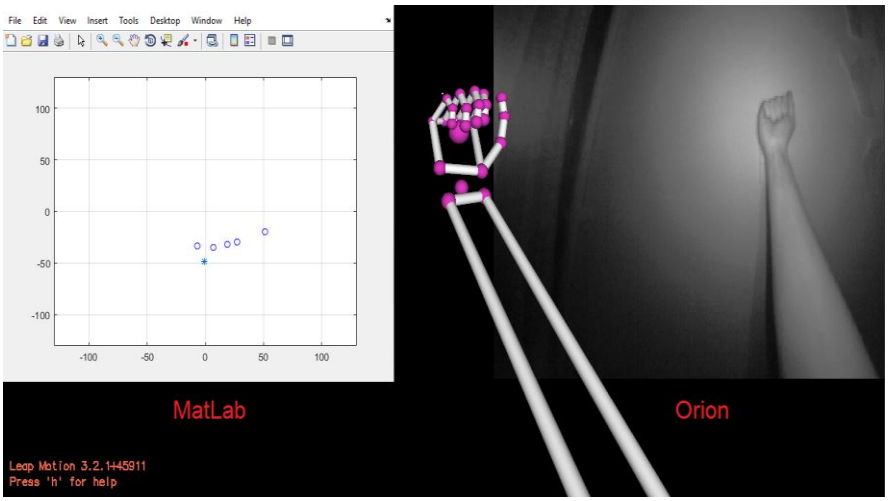


Figura 16. Visualización de la seña A en tiempo real, a la izq. en MatLab, a la der. en "Orion".
Fuente: Elaboración Propia.

Las señas representadas en MatLab, a diferencia a las señas en "Orion", no son fáciles de identificar, ya que los puntos representados por las yemas de los dedos se distribuyen en distancias muy cercanas entre sí, generando similitudes entre las señas, un caso concreto son las señas A y E, donde la diferencia radica en la posición del dedo pulgar, en la *seña A* el dedo pulgar es el que tiene mayor altura en el plano cartesiano, y los demás dedos quedan debajo de la palma, y en la seña E el pulgar es el que tiene menor altura en el plano cartesiano, y los demás dedos quedan por encima de la palma, como se presenta en la Figura 17.

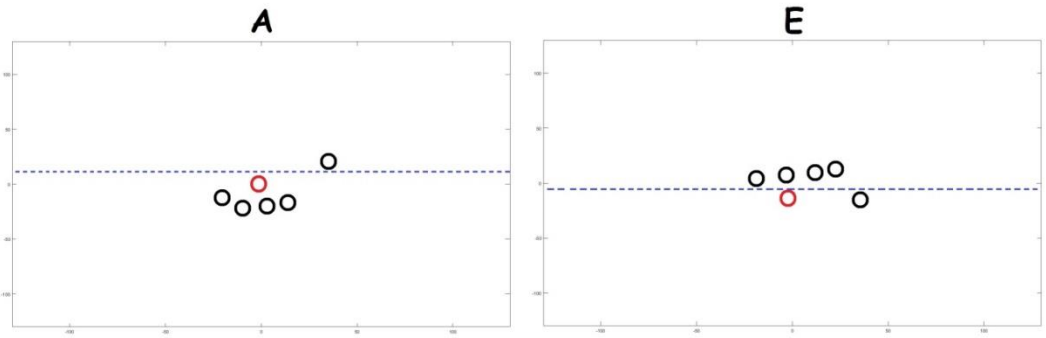


Figura 17. Comparación entre la seña A con la seña E en MatLab.
Fuente: Elaboración Propia.

Para facilitar la visualización de los puntos en MatLab se resaltó con colores las yemas de los dedos: meñique con el color verde, anular con el color rojo, medio con el color amarillo, índice con el color azul, el dedo pulgar con el color rosado y el color blanco para la identificación de la palma. Figura 18.

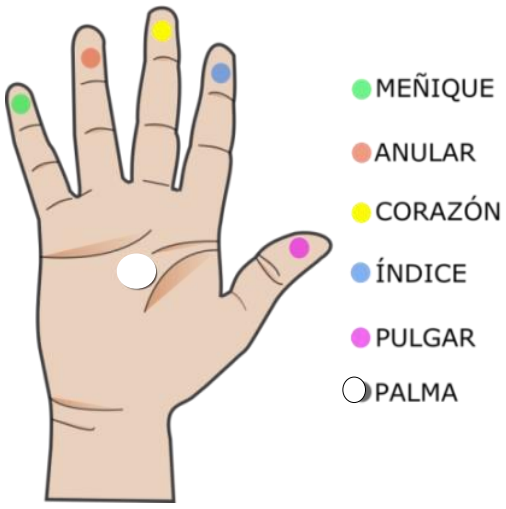
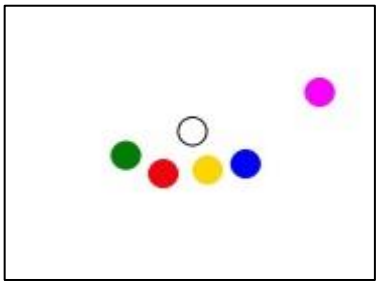
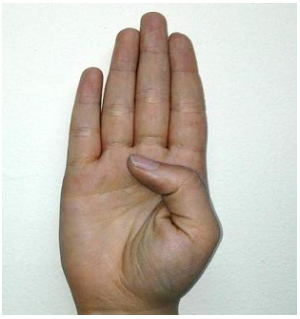
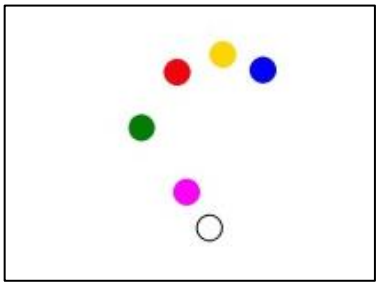
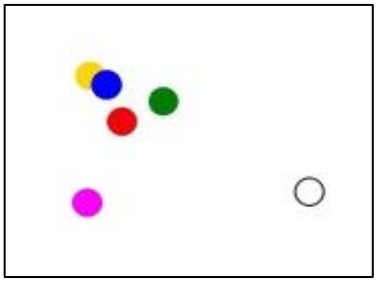
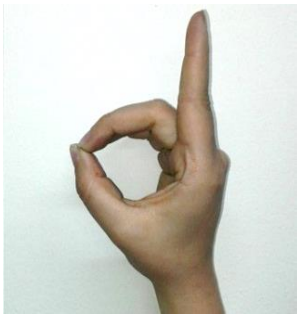
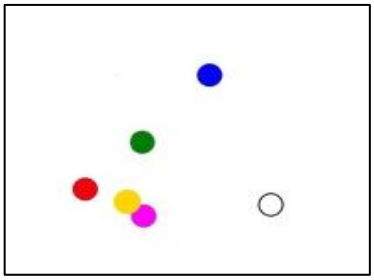
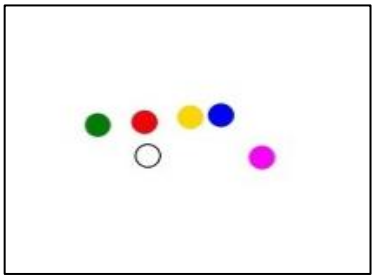
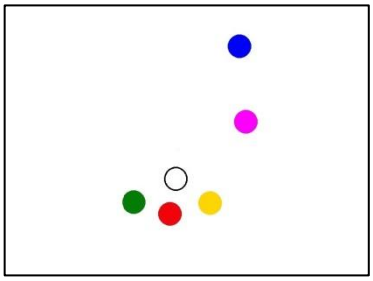
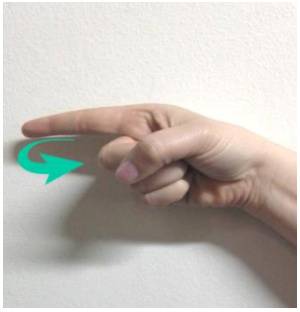
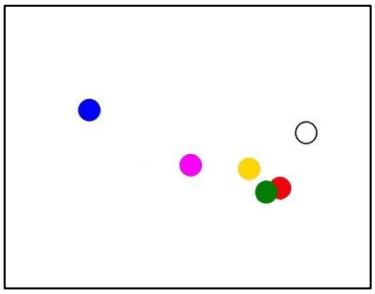
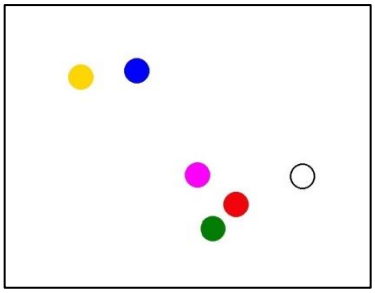


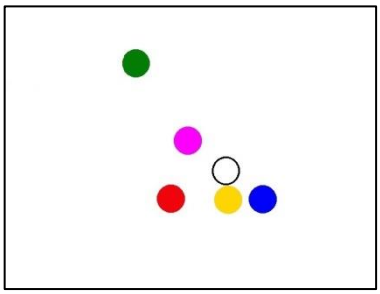
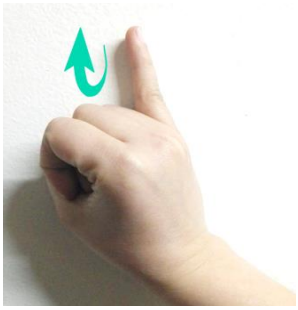
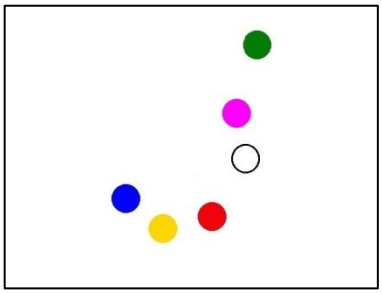
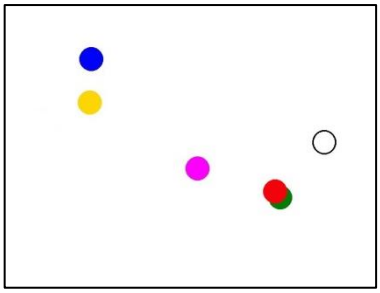
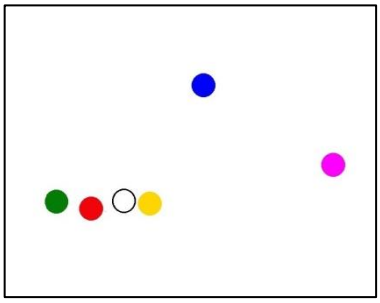
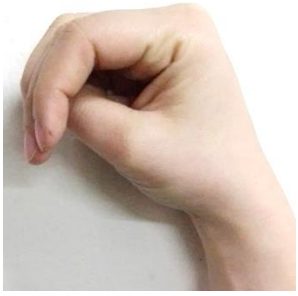
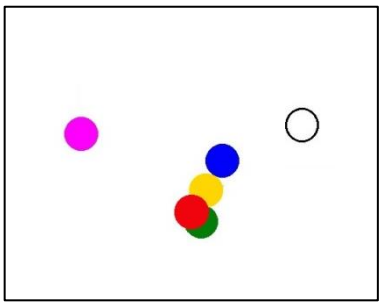
Figura 18. Identificación de las yemas de los dedos con colores para la identificación en MatLab

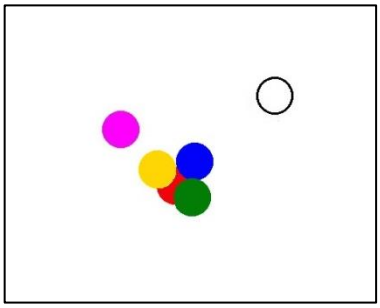
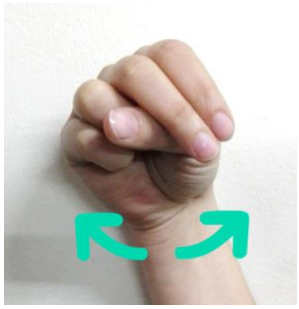
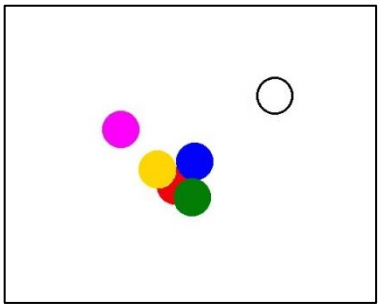
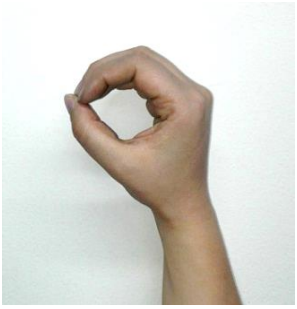
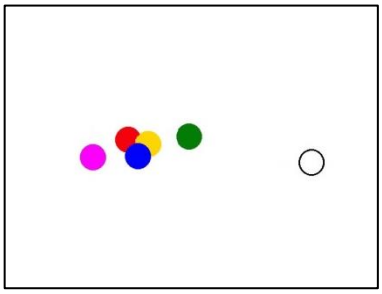
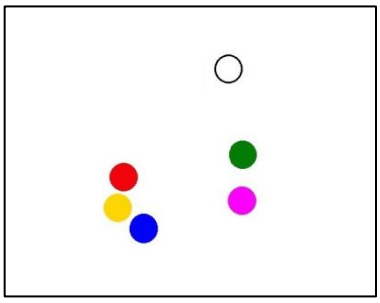
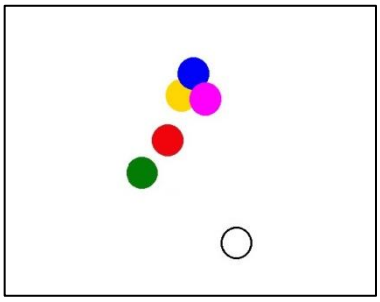
Teniendo la identificación de las yemas de los dedos por medio de colores, se procede a realizar un contraste del alfabeto de la LSC con la representación de las señas en MatLab, como se presenta en la Tabla 2.

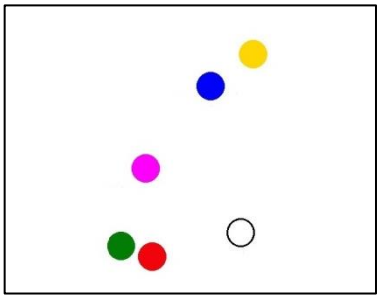
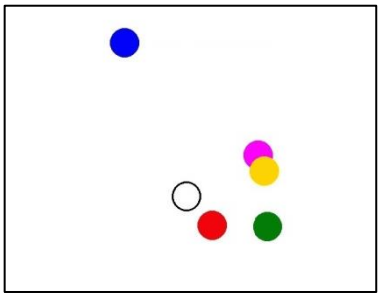
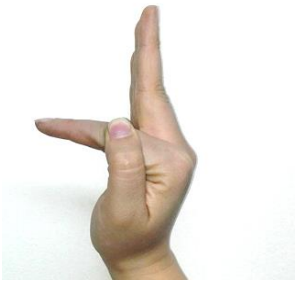
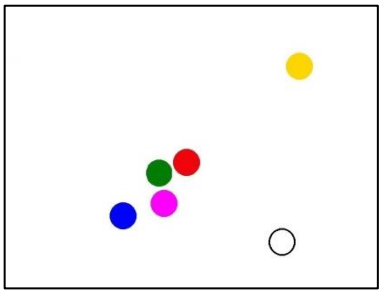
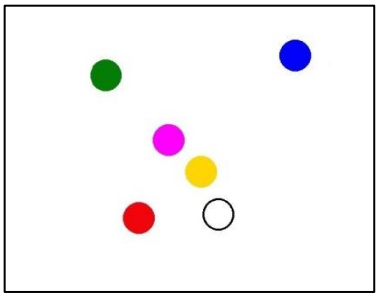
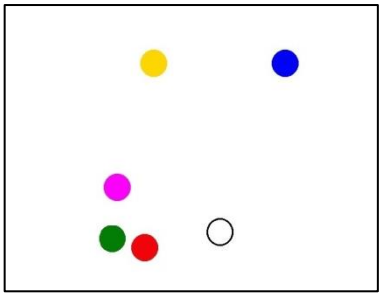
Tabla 2. Contraste del Alfabeto de la LSC real con las señas realizadas en MatLab.

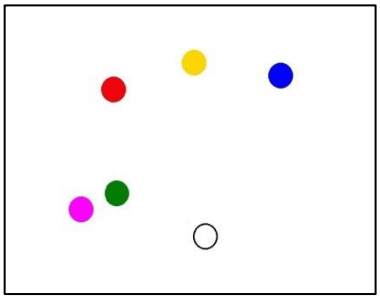
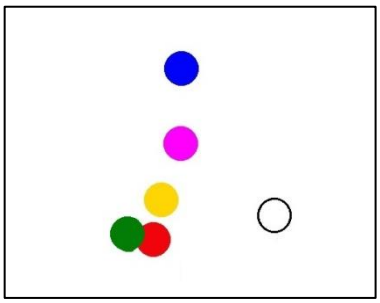
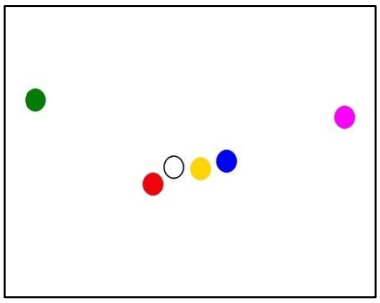
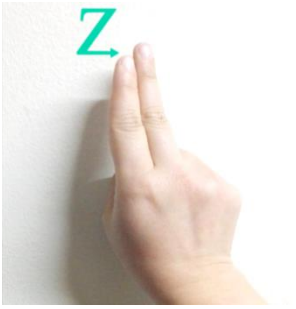
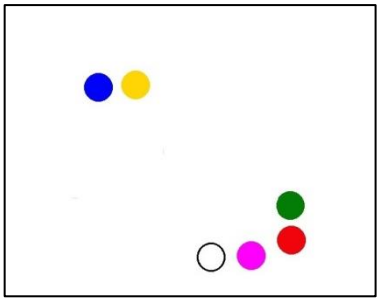
LETRA	SEÑA REAL	SEÑA EN MATLAB	DESCRIPCIÓN
A			La seña A visualizada en MatLab, se asemeja a la seña real, y se puede diferenciar de las demás señas visualizadas en MatLab, porque la posición del dedo pulgar se encuentra por encima de la palma de la mano y los otros dedos están por debajo de la palma.
B			La seña B visualizada en MatLab, se parece a la seña real, además se diferencia de las otras señas realizadas en MatLab, ya que la posición de los dedos es visible para la lectura del Leap Motion.
C			La seña C visualizada en MatLab, presenta una agrupación de los dedos meñique al índice, que no se asemeja a la posición de los dedos en la seña real.

D			La visualización de la seña D en MatLab no se asemeja a la seña real, porque en el gesto real el dedo medio cubre al anular y meñique, lo cual genera una obstrucción visual para que el sensor Leap Motion de una posición precisa de los dedos.
E			La seña E tiene similitud a la seña A visualizada en MatLab, por las posiciones de los puntos verde al azul, pero en la seña E cambia por la posición del dedo pulgar y la palma de la mano, ya que se encuentran debajo de los demás dedos.
F			La seña F visualizada en MatLab, se parece a la seña real, y se diferencia de las demás señas simuladas, por la posición de los dedos índice y pulgar.
G			La seña G se tomó estática para poder tomar las posiciones de los dedos, de allí, se observó que la visualización en MatLab se asemeja a la seña real.
H			La seña H se tomó estática para poder observar las posiciones de los dedos, de allí, se observó que la visualización en MatLab se asemeja a la seña real.

I			La seña I visualizada en MatLab, se parece a la seña real y se diferencia de las otras señas simuladas por la posición del dedo meñique.
J			La seña J se tomó estática para poder observar las posiciones de los dedos, se puede diferenciar de las demás señas visualizadas en MatLab por la posición del dedo meñique.
K			La seña K visualizada en MatLab, no tiene similitud con la seña real, porque el punto amarillo que representa el dedo medio está ubicado cerca al dedo índice.
L			La seña L visualizada en MatLab, se puede diferenciar de las demás señas simuladas, por la posición de los dedos índice y pulgar, además se asemeja a la seña real.
M			La seña M visualizada en MatLab, se diferencia de las otras señas simuladas por la posición del punto rosado (dedo pulgar), aun así, no se asemeja a la seña real.

N			La seña N visualizada en MatLab, se confunde con la simulación de la seña M, por las posiciones de los dedos, la única diferencia está en la ubicación del punto rosado (pulgar).
Ñ			La seña Ñ se tomó estática para poder observar las posiciones de los dedos en MatLab, y por tal razón se confunde con la simulación de las señas M y N.
O			La seña O y la seña C tienen cierta similitud en la visualización de MatLab, porque presenta una agrupación de los dedos meñique al índice, la única diferencia es que el dedo pulgar está pegado a los demás dedos, en cuanto a la seña real no tienen similitud.
P			La seña P visualizada en MatLab se puede diferenciar de las demás simulaciones por la posición de la muñeca, lo cual permite una distribución de los dedos diferente a las demás señas.
Q			La seña Q visualizada en MatLab, se asemeja a la seña real por la agrupación de los dedos.

R			La seña R visualizada en MatLab, se asemeja a las posiciones de los dedos de la seña real.
S			La seña S se tomó estática para poder observar las posiciones de los dedos, de allí, se observó que la visualización en MatLab se asemeja a la seña real.
T			La seña T visualizada en MatLab no se parece a la seña real, porque la lectura no es la ideal por el Leap Motion, ya que los dedos anular y meñique en la simulación deberían estar en una posición próxima al dedo medio.
U			La seña U simulada en MatLab se asemeja a las posiciones de los dedos en la seña real. Además, se diferencia de las demás señas simuladas por las posiciones de los puntos verde y azul.
V			La seña V simulada en MatLab se asemeja a las posiciones de los dedos en la seña real. Además, se diferencia de las otras señas simuladas por las posiciones de los puntos amarillo y azul.

W			La seña W simulada en MatLab se asemeja a las posiciones de los dedos en la seña real. Además, se diferencia de las otras señas simuladas por las posiciones de los puntos rojo, amarillo y azul.
X			La seña X visualizada en MatLab, se asemeja a la seña real, teniendo en cuenta que la mano está cerrada en la seña real.
Y			La seña Y visualizada en MatLab, se asemeja a la seña real.
Z			La seña Z se tomó estática para poder observar las posiciones de los dedos, de allí, se observó que la visualización en MatLab se asemeja a la seña real.

3.2.2 Captura de la LSC en MatLab.

Para la captura de datos del alfabeto de LSC, primero se selecciona la mano con la cual se va a trabajar, para este caso la mano derecha y de muestra se tomó la mano de una

persona para capturar las posiciones espaciales de las señas del alfabeto, luego se realizó un algoritmo llamado `build.m`[14] que permitió capturar las posiciones de las yemas de los dedos, para esto, se definió el área de trabajo y la profundidad de detección del sensor, logrando tener control del campo visual y la distancia entre las manos y la frente del usuario, luego se realizó el algoritmo principal de la aplicación, que consistió en la resta de la posición cartesiana de la palma con la posición cartesiana de cada dedo, con el fin de mantener las distancias espaciales de cada seña y así poder ser identificada en cualquier parte del plano cartesiano. Cabe resaltar que el sensor Leap Motion se adapta a los tamaños de las manos que detecta, por lo tanto, no es necesario tomar muestras de diferentes manos.

Para ejemplificar el proceso de captura de datos de las señas se tomó como muestra la seña A visualizada en MatLab, de este modo se toma como referencia la posición cartesiana del punto blanco que representa la palma y se resta con la posición cartesiana del punto verde que representa el dedo meñique, Figura 19.

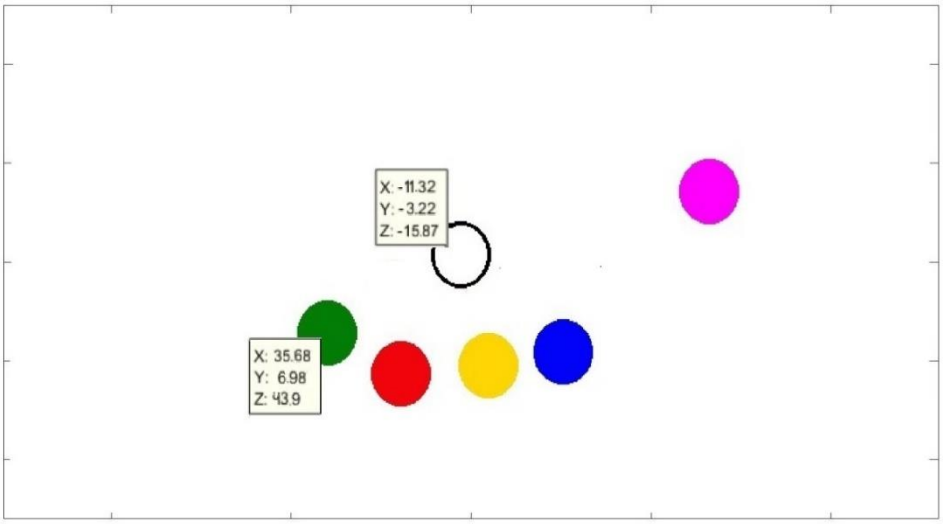


Figura 19. Representación de la seña A, indicando las posiciones cartesianas de la palma y la yema del dedo meñique. Fuente: Elaboración Propia.

[14] Es un script que permite extraer los datos de posición, velocidad, frames, que detecta el sensor Leap Motion y los muestra en MatLab.

Las posiciones del sistema cartesiano que entrega MatLab de la palma y de las yemas de los dedos, se restan punto a punto, como se presenta en la Tabla 3, y el resultado generado se almacena en una matriz llamada matriz de entrada.

Tabla 3. Diferencia entre las posiciones cartesianas entre la Palma y la yema del dedo Meñique.

Distancia entre la Palma y el dedo Meñique			
Objeto/Posición	X	Y	Z
Posición Palma	-11,324	-3,221	-15,875
Posición Meñique	35,681	6,9821	43,948
Resultado	24,357	3,7611	28,073

Cuando se termina de capturar las posiciones, se procede a ubicar en la matriz de entrada todos los resultados obtenidos empezando por los datos del dedo pulgar (X1, Y1, Z1), índice (X2, Y2, Z2), medio (X3, Y3, Z3), anular (X4, Y4, Z4) y meñique (X5, Y5, Z5), como se presenta en la Tabla 4.

Tabla 4. Datos de las posiciones de los dedos en X, Y, Z de la seña A.

DATOS DE ENTRADA LETRA A															
LETRA	PULGAR			ÍNDICE			CORAZÓN			ANULAR			MEÑIQUE		
	(X1)	(Y1)	(Z1)	(X2)	(Y2)	(Z2)	(X3)	(Y3)	(Z3)	(X4)	(Y4)	(Z4)	(X5)	(Y5)	(Z5)
A	-44,37	-29,06	15,339	-17,25	1,9849	40,44	-3,938	6,646	38,276	11,398	9,4236	33,713	24,357	3,7611	28,073

El procedimiento que se mencionó anteriormente se repitió con las 27 letras del alfabeto colombiano, obteniendo un tamaño de la matriz de entrada de *81X27*.

Cabe resaltar, que en la LSC existen señas con movimiento (G, H, J, Ñ, S y Z), por lo tanto, la estrategia de trabajo fue extraer los datos de forma estática, para tener una matriz con la misma cantidad de datos por cada seña.

Continuando con el ejemplo de la seña A, se toma como base las posiciones de las vocales, teniendo una matriz de entrada con un tamaño de 15*x5* (Filas y Columnas), las filas presentan la cantidad de posiciones cartesianas espaciales tomadas por la diferencia de la posición de la palma de la mano con cada dedo, las columnas presentan la cantidad de señas (las vocales), como se presenta en la Figura 20.

DATOS DE ENTRADA VOCALES

LETRAS	PULGAR			ÍNDICE			CORAZÓN			ANULAR			MEÑIQUE		
	(X1)	(Y1)	(Z1)	(X2)	(Y2)	(Z2)	(X3)	(Y3)	(Z3)	(X4)	(Y4)	(Z4)	(X5)	(Y5)	(Z5)
A	-44,37	-29,06	15,339	-17,25	1,9849	40,44	-3,938	6,646	38,276	11,398	9,4236	33,713	24,357	3,7611	28,073
	-65,19	-16,99	-26,13	-39,53	3,5621	1,9045	-33,11	5,4185	12,679	-21,28	10,444	21,726	-10,01	7,1473	29,951
	-23,49	-29,14	62,242	2,2738	2,949	36,448	14,637	5,5875	29,535	24,242	8,6597	20,658	32,44	5,3964	9,3501
E	-37,83	-8,797	48,573	7,1451	-23,96	39,296	22,165	-23,08	30,691	-22,86	-24	39,039	-10,84	-27,16	43,532
	-79,63	9,8811	-17,13	-42,81	-18,04	13,478	-27,56	-3,315	24,975	-13,13	-5,696	31,987	2,2884	-10,47	33,924
	-4,149	-16,42	58,411	-5,038	-78,39	47,599	1,3121	-92	34,537	14,712	-88,04	12,894	26,284	-71,81	-11,98
I	27,387	-67,2	14,56	14,597	-24,4	38,737	-12,87	3,9365	37,385	0,3202	4,8685	35,995	14,446	6,7346	29,703
	6,9644	-49,87	43,613	9,904	-101,4	37,045	-11,38	10,913	42,043	-20,59	15,989	33,027	-32,01	13,781	20,754
	28,417	-26,85	44,506	31,301	-6,658	28,643	33,443	1,0635	18,711	36,422	1,2031	5,7886	34,307	-2,731	-9,343
O	60,004	-10,34	22,541	52,367	-13,28	15,708	48,862	-5,266	1,8904	41,287	-2,901	-13,46	32,867	-7,603	-25,31
	56,106	-32,43	6,3344	46,296	-28,02	-10,74	46,697	-37,62	45,072	45,891	-42,97	34,774	56,07	-40,8	25,005
	37,472	-14,53	-33,87	20,086	-14,59	-37,72	56,084	-25,28	-3,552	57,098	-25,84	-13,45	55,022	-27,07	-23,8
U	-23,55	-79,04	3,0613	28,395	-57,97	10,212	10,64	-32,61	32,604	-0,989	-8,741	34,783	11,757	-4,383	29,815
	-40,77	-93,15	-30,63	7,117	-83,54	21,273	-16,82	-32,24	38,768	-27,55	8,535	31,607	-15,23	8,2876	37,539
	-6,582	-87,54	13,941	29,674	-0,851	-16,06	34,802	-35,6	14,418	35,651	-5,006	10,763	33,962	2,2316	-1,345

Posiciones de cada dedo (15)

Cantidad de señas (5)

Figura 20. Matriz de entrada de las posiciones cartesianas de las Vocales.
Fuente: Elaboración Propia.

Con los datos obtenidos de entrada se crea una matriz con datos de salida correspondiente a cada letra con el fin de establecer una relación entre las matrices. La función de la matriz de salida es activar un bit para la identificación de cada seña. La matriz cuenta con un tamaño de *5X5* (Filas y Columnas), las filas se refieren a la activación de un 1 en la cadena de bits y las columnas corresponden a la cantidad de señas (las vocales), como se presenta en la Figura 21.

Figura 21. Matriz de salida que se relaciona directamente con la matriz de entrada.
Fuente: Elaboración Propia.

3.2.3 Validación de las Redes Neuronales de la LSC en MatLab.

Para validar y comprobar las señas del alfabeto colombiano, se usa el Toolbox de MatLab llamado *Neural Network*, el cual proporciona varias alternativas para trabajar con redes neuronales, el objetivo de esta herramienta es que por medio de diferentes entrenamientos la RNA aprenda y reconozca los datos que se le ingresa al sistema. En este proyecto se usó la red neuronal *feed-forward*, la cual está conformada de una capa de

entrada, una capa de salida y una capa oculta, luego se agregan las matrices de entrada y salida, explicadas en el apartado anterior, estas matrices se deben relacionar por medio de filas, luego se procede a la arquitectura de la Red Neuronal, allí se encuentra una cantidad definida de neuronas en la capa oculta, esta opción se puede modificar por el usuario para establecer una cantidad apropiada, con el fin de hallar un buen resultado en el entrenamiento, como se presenta en la Figura 22.

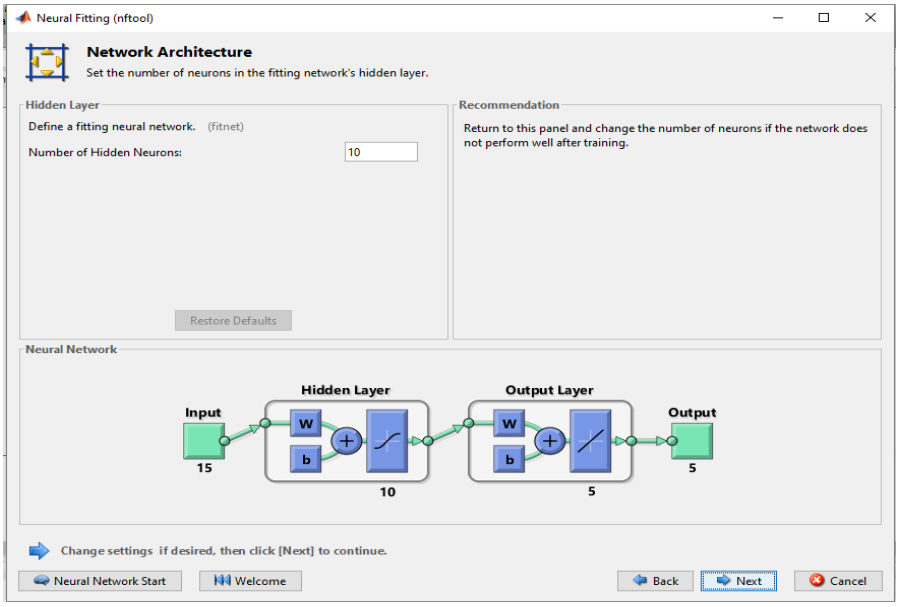


Figura 22. Estructura de la RNA y definición de la cantidad de neuronas en la capa oculta.
Fuente: Elaboración propia.

Por último, se selecciona el algoritmo de entrenamiento, para este proyecto se usaron dos métodos el *Levenberg Marquartdt*, el cual tiene como ventaja la ejecución de los entrenamientos sin gastar tanto los recursos del computador, sin embargo la desventaja es el funcionamiento limitado al entrenar una cantidad pequeña de datos, y la *Regularización Bayesiana*, que tiene como ventaja permitir trabajar con problemas o ejercicios que contienen una mayor cantidad de datos, sin embargo esto ocasiona una

desventaja ya que gasta más los recursos del computador porque en cada entrenamiento actualiza los valores de *peso* y *bias*.

De acuerdo con los resultados obtenidos en los dos tipos de entrenamientos, se observó que la *Regularización Bayesiana* se acomodó a las necesidades del proyecto porque se acercó al tipo de entrenamiento ideal y como resultado se dio un *error cuadrático medio (MSE)* cercano a 0 y una *regresión* con un valor cercano a 1, características ideales del mejor entrenamiento del proyecto, como se muestra en la Figura 23.

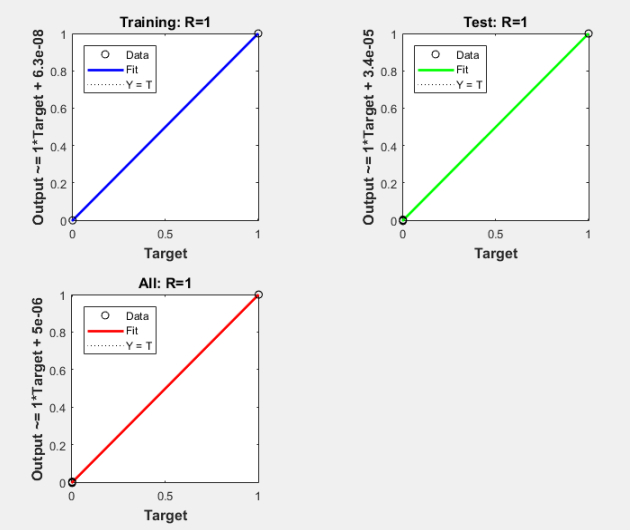


Figura 23. Visualización del entrenamiento Bayesiano ideal de la Red Neuronal. Fuente: Elaboración propia.

Así mismo, los resultados de los entrenamientos se almacenaron en dos Redes Neuronales, la primera RNA con el entrenamiento *Levenberg Marquartdt* y la segunda RNA con el entrenamiento *Regularización Bayesiana*, las cuales fueron evaluadas por separado con la función `sim=(RedEntrenada, DatosTiempoReal')` que compara los datos entrenados por la Red con los datos capturados por el Leap Motion en tiempo real.

Los resultados de validación se vieron afectados por las posiciones de los dedos, a causa de las similitudes en algunas de las señas, como en el caso de la seña A con la E, donde la RNA *Levenberg Marquartdt* tuvo un acierto del 70% en la identificación de cada una de las señas, es decir, tuvo momentos en los que la RNA confundía la seña A con la seña E, y en el caso de la RNA *Regularización Bayesiana*, se obtuvo un acierto del 80%, como se presenta en la Figura 24, además en las señas M, N y Ñ, se obtuvo un resultado de acierto del 50% en la RNA con el entrenamiento *Levenberg Marquartdt* y un 60% de acierto en la RNA con el entrenamiento *Regularización Bayesiana*, debido a que el sensor Leap Motion no tiene una visualización completa de las posiciones de las yemas de los dedos, porque estas señas se realizan con la mano cerrada.

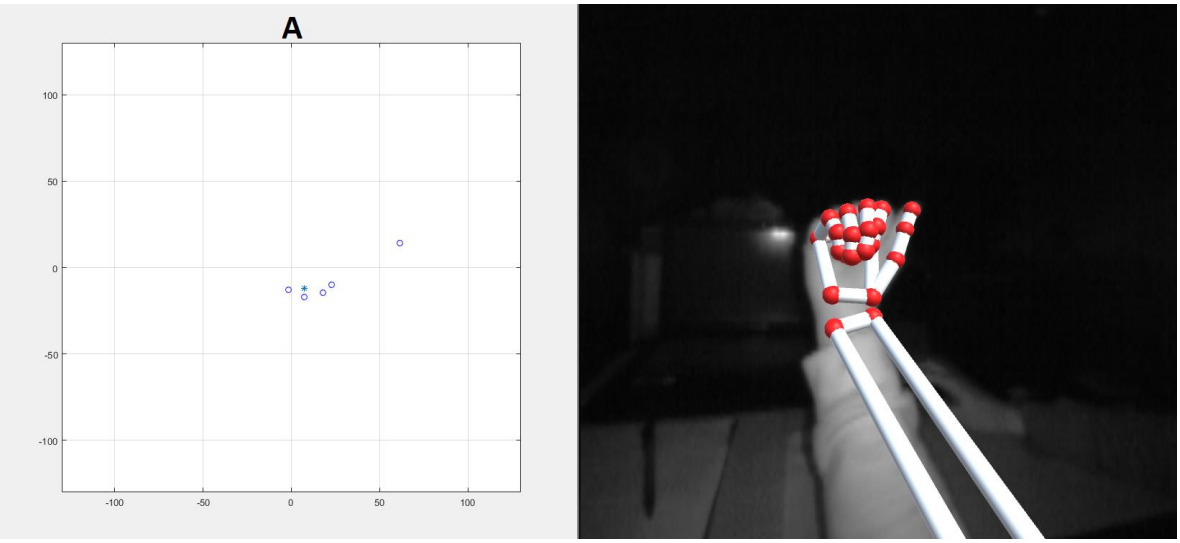


Figura 24. Comprobación de la RNA entrenada en tiempo real: izq. Visualización de la seña A en MatLab, y der. Visualización de la seña en "Orion". Fuente: Elaboración Propia.

Se puede observar en la Tabla 5, la toma de 10 muestras por cada seña, que da como resultado un 68,8% de acierto con el algoritmo de entrenamiento *Levenberg Marquartdt* y un 83,7% de acierto con el algoritmo *Regularización Bayesiana*.

Tabla 5. Comparación entre las dos Redes Neuronales de entrenamiento con los respectivos porcentajes de acierto y error en cada seña realizada. Fuente: elaboración Propia.

ALFABETO	Levenberg Marquartdt		Bayesian Regularation		CONTRASTE
	ACIERTO (%)	ERROR (%)	ACIERTO (%)	ERROR (%)	
A	80	20	80	20	E
B	90	10	90	10	
C	80	20	100	0	
D	80	20	90	10	
E	70	30	80	20	A
F	70	30	80	20	
G	60	40	70	30	
H	50	50	60	40	Z,K
I	80	20	100	0	
J	90	10	90	10	
K	80	20	80	20	H,Z
L	70	30	100	0	
M	50	50	60	40	N,Ñ
N	50	50	60	40	M,N
Ñ	50	50	60	40	N,M
O	80	20	90	10	
P	50	50	80	20	
Q	80	20	90	10	
R	60	40	100	0	
S	80	20	80	20	
T	50	50	80	20	
U	70	30	90	10	V
V	70	30	90	10	U
W	80	20	100	0	
X	50	50	80	20	
Y	80	20	100	0	
Z	60	40	80	20	H,K

Como resultado, se seleccionó la RNA con el algoritmo de entrenamiento *Regularización Bayesiana*, debido a que tuvo mayor porcentaje de acierto en la identificación del alfabeto de la LSC.

Por último, se procedió a realizar un algoritmo que permite la comunicación entre MatLab y Unity, la cual se diseñó para el transporte en tiempo real de los datos capturados por el Leap Motion en contraste a los obtenidos en la Red neuronal, el resultado de los datos obtenidos se almacenan en un archivo *.txt* a través de la función `fid = fopen('E:/Assets/RE.txt','w')` que es escrito por MatLab, y Unity se encarga de leer los datos escritos en el archivo *.txt* a través de un algoritmo que evalúa en tiempo real si la seña realizada por el usuario es la correcta .

3.3 Sprint III: Diseño y desarrollo de la aplicación en Unity.

En este sprint se trabajó con el software Unity, el cual proporcionó el aspecto visual y el ambiente virtual del proyecto, tomando como referencia el uso de las señas del alfabeto. En el proceso de trabajo el primer paso fue diseñar los escenarios, los cuales fueron unificados en un salón de clase para generar un espacio educativo, luego se diseñaron las mecánicas de interacción, las cuales determinan las acciones que puede realizar el usuario estando inmerso en la aplicación y por último se explica la programación o algoritmos que fueron usados para el funcionamiento de las actividades de la aplicación.

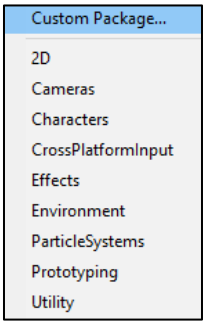
3.3.1 Estructura del proyecto

Para empezar a hablar del diseño y desarrollo de la aplicación, es importante aclarar que el proyecto fue desarrollado gracias a los siguientes sistemas: *Unity versión 2017.3.1f1, Sensor Leap Motion (Orion versión de software 4.0.0+52173, Core versión 4.0.0 para Unity) y MatLab R2015b.*

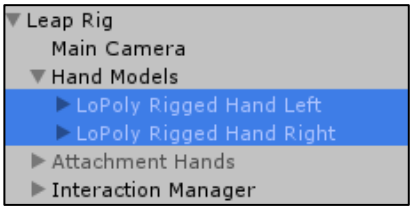
En Unity se diseñaron varias herramientas de trabajo que van ligadas a la parte del funcionamiento y presentación gráfica del proyecto, estas herramientas van desde la implementación del sensor Leap Motion en Unity, el diseño de los objetos en los escenarios hasta la configuración para trabajar en Realidad Virtual en Unity, como se presenta en la Tabla 6.

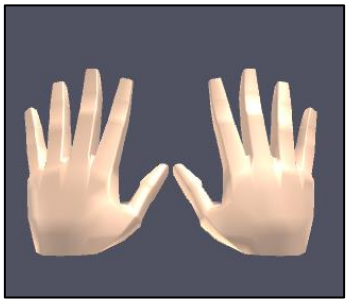
Tabla 6. Componentes del proyecto en Unity.

<table>
<tr><th>Carpetas</th><th>Subcarpetas</th><th>Aplicación</th></tr>
<tr><td rowspan="2">Programas: En esta carpeta se localizan las subcarpetas que contiene los SDK “Software Development Kit” por sus siglas en inglés y que en español significa “Kit de desarrollo de software”,

Los SDK mencionados anteriormente se usaron para desarrollar la aplicación en realidad virtual con ayuda del Leap Motion en Unity.</td><td>Editor: En esta carpeta se guardan los diferentes Package importados de Unity, que fueron importantes para diseñar la interfaz gráfica de la aplicación.</td><td> 
</td></tr>
<tr><td>SDK de LeapMotion:

“Permite la presencia de las manos en la RV, gracias al SDK, Assets Core y Modules que facilitan el diseño de las manos, interfaces e interacciones en Unity.”

https://developer.leapmotion.com/unity/#5436356</td><td>Hand Models: A esta carpeta se le agregaron los dos Prefabs de las manos con las que el usuario interactúa en los diferentes escenarios en la realidad virtual.
 
</td></tr>
</table>

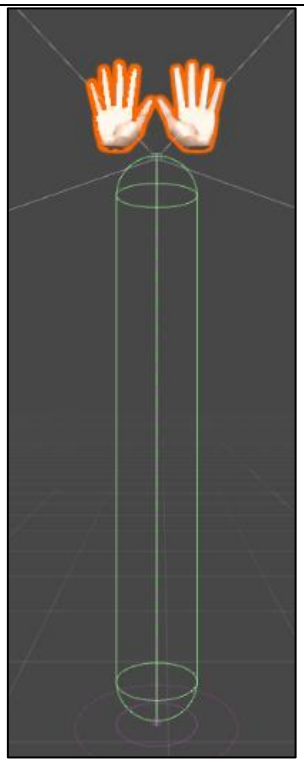
<table>
<tr>
<td rowspan="2"> 
</td>
<td> 
</td>
<td>La textura de las manos fue dada por el color piel en hexadecimal #fdddca.
  </td>
</tr>
<tr>
<td>SDK de Oculus:

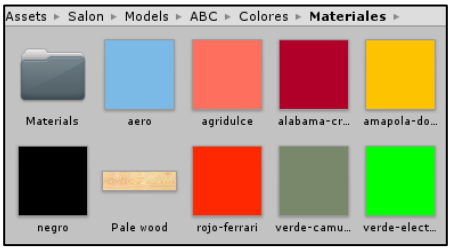
Permite la relación entre las gafas Oculus y Unity, ya que el SDK contiene los recursos (gráficos, cámaras, scripts, entre otros), que fueron importantes para el diseño de esta aplicación.

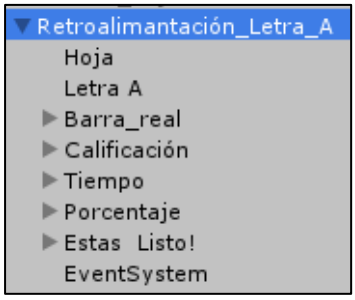
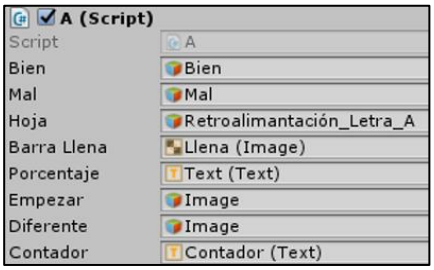
 
</td>
<td>El Asset OVRPlayerController: Es un control prefabricado que permite al usuario ser un personaje en primera persona en la realidad virtual, a este Asset se le agregó el Asset Leap Rig: que contiene todos los recursos para usar las manos en la realidad virtual.

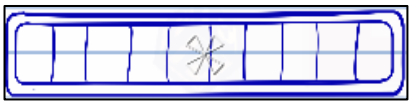
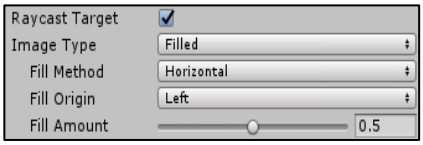
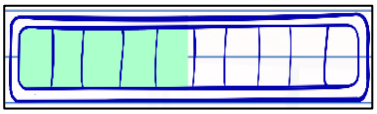
 

Vista del Prefab en modo desarrollador: Asset OVRPlayerController y Asset Hand Models</td>
</tr>
</table>

<table>
<tr><td rowspan="3">Aplicación: En esta carpeta se guardan todas las carpetas que se usaron en el diseño de la aplicación.</td><td></td><td>  </td></tr>
<tr><td>Escenas: Aquí se guarda la introducción, el menú y las actividades.</td><td> 
</td></tr>
<tr><td>Imágenes: Se guardan las imágenes que le dan un aspecto atractivo a la experiencia.</td><td> 
</td></tr>
</table>

	Materiales: Aquí se encuentran todas las imágenes y colores que dan realismo a los escenarios como: la textura del piso, techo, paredes entre otros.	
	Prefabs: Son los objetos que son diseñados, creados y reutilizados como: cubos, paredes, ventanas, pisos, sillas, tablero, etc.	

	Scripts: Se guardan los códigos que se usaron en la comunicación entre Unity y MatLab, y en la interacción de los objetos que se encuentran en las escenas.
	
	Hay 27 GameObject Retroalimentación_Letra_A a la Z, a los que se le agregó el Script A.cs.
	
	 
		(Script): A.cs
		
		 
		
		`LeerArchivo ():`
		Esta función busca la *path o ruta* donde se creó el archivo.txt que *MatLab* escribe en tiempo real. Cuando se encuentra esta Path Unity decodifica el mensaje de *MatLab* escribe y los guarda en la variable llamada *textoMatLab.*
		
		`if (Hoja.activeSelf) {…}:`
		esta parte del código se encuentra esperando si la *hoja_Letra_A* ha sido activada, en el caso en que esta hoja este activa, inmediatamente aparecerá un aviso que dice *Listo* y

		transcurrido unos segundos aparecerá un contador y una barra. `BarraLlena.fillAmount=textoMatLab:` Esta función cambia el tamaño del *FillAmount* que hace parte del tipo de imagen, además permite cambiar el aspecto de lleno y vacío de la barra, y cambia dependiendo de lo que se está almacenando en la función *textoMatLab.*   *FillAmount = 0, Barra = vacía*   *FillAmount = 0.5, Barra = se llena hasta la mitad* **`contador.text = " " + tiempo.ToString("f0");`** Esta función siempre revisa si las señas son realizadas y si corresponde a lo que está escribiendo MatLab, si es así califica la seña escribiendo el porcentaje del acierto o desacierto de la seña realizada y lo muestra encima de la barra (0 – 100%).

	Vídeos: En esta carpeta se guardaron los *videosClip* que fueron grabados y editados previamente, para ser utilizados en la aplicación.  	**Videos de Instrucciones:** Se guardan dos videos de instrucciones que fueron agregados al escenario menú Práctica y Prueba. **Videos de Señas:** Se encuentran almacenados los 27 videos de las señas de la LSC que se agregaron al escenario práctica.

3.3.2 Mecánicas de interacción

En seguida se hablará de los objetos dinámicos que se encuentran en el proyecto (imágenes y videos), se les denomina dinámicos ya que poseen algún tipo de animación, acción e interacción en los escenarios o con el usuario. Para describirlos uno a uno se realizó en la Tabla 7 una explicación sobre la ubicación del objeto en el escenario, la descripción de este y el script que le proporciona las propiedades que permiten interactuar con él.

Tabla 7. Objetos dinámicos del proyecto.

<table>
<tr><td colspan="3">Escenario: Menú.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual.</td></tr>
<tr><td colspan="3">Nombre: Botón play azul.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td> 
Ilustración 1. Fuente: Adaptado de https://hubpages.com/art/Createaglossyroundwebbuttoninginp28</td><td>El Botón play azul es un UI Image al que se le agregó la imagen que es asociada con el símbolo de Reproducir.</td><td>InteractionButton.cs
El script proporciona un conjunto de interacciones físicas, que permiten presionar el botón.

Para el proyecto se editó el script para que, al ser presionado el botón, se active la reproducción de los videos, los cuales indican las instrucciones que el usuario debe seguir en los escenarios: Práctica y Prueba.</td></tr>
<tr><td colspan="3">Escenario: Menú.</td></tr>
<tr><td colspan="3">Nombre: Botón Empezar Práctica y Prueba</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td> 
Ilustración 2. Fuente: Elaboración Propia.</td><td>Las imágenes de práctica y prueba son componentes de tipo UI aquí se agregó la imagen que contiene los nombres de las actividades, estos al ser presionados activan los escenarios: Práctica y Prueba.</td><td>InteractionButton.cs
Este script se agregó a los dos botones Práctica y Prueba, para proporcionar la interacción con estos y así poder ser presionados por el usuario.

A continuación, se editó este script, para que, al presionar los botones, el usuario ingrese a los escenarios: Práctica y Prueba.</td></tr>
</table>

<table>
<tr><td colspan="3">Escenario: Menú.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual.</td></tr>
<tr><td colspan="3">Nombre: Videos de instrucciones Práctica y Prueba.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td> 
</td><td>Los videos son 3D Object>Plane de Unity, al Plane se le agregó un Add Component llamado Video Player, este componente permitió agregar los dos videoclips, que fueron grabados previamente explicando lo que se debe hacer en cada uno de los escenarios: Práctica y Prueba</td><td>InteractionButton.cs

Este script se agregó al Plane para proporcionar la interacción con el botón play.

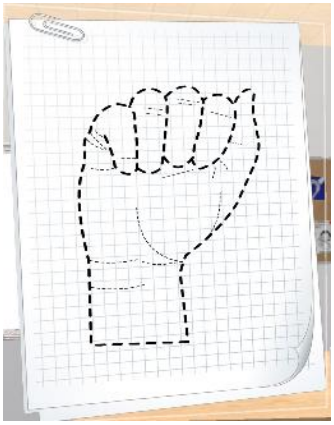
OnPress (). Esta función se activa si el botón ha sido presionado, al instante se activará el Videoclip y desactivará el botón play.</td></tr>
<tr><td colspan="3">Escenario: Práctica.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual y observación de los movimientos de las señas por medio de los videos.</td></tr>
<tr><td colspan="3">Nombre: Videos de las 27 letras del alfabeto de la LSC.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td>  </td><td>Los videos son 3D Object>Plane de Unity, al Plane se le agregó un Add Component llamado Video Player, este componente permitió agregar los 27 videoclips, que fueron grabados previamente explicando cómo se debe realizar las señas de la LSC</td><td>InteractionButton.cs

Este script se agregó a cada uno de los 27 CubeButton, ya que les proporciona la interacción con cada uno de los botones a los cuales corresponde.

A este script se le editó la función ContactBegin (). Esta parte del código siempre pregunta si el CubeButton, “A” ha sido presionado, si se presionó activa el videoclip A y desactiva los 26 CubeButton restantes.</td></tr>
</table>

Escenario: *Práctica.*		
Actividad: *Exploración del escenario en la Realidad Virtual e interacción con los botones del alfabeto.*		
Nombre: *Botones de las 27 señas del alfabeto.*		
Objetos	*Descripción*	*Script*

	Los botones están compuestos de dos cubos sobrepuestos, el cubo que esta atrás tiene la función de base, el que está en frente sirve para darle un aspecto estético que el desarrollador quiera, en este caso se le agregaron a cada uno de los 27 *CubeButton* las imágenes del alfabeto que previamente fueron diseñadas en *Inskape.*	`InteractionButton.cs`
		
		Este script se agregó a cada uno de los 27 *CubeButton,* para proporcionar la interacción correspondiente a cada uno del botones.
		
		A modo de ejemplo del resultado de editar el script se observa que al presionar el *CubeButton* con la letra “A” se reproduce el video correspondiente a la seña en la LSC. Pasados unos segundos se activa una hoja de cuaderno para que el usuario practique la seña “A”.
		
		`SimpleInteractionGlow.cs`
		
		Este script facilita la propiedad de cambiar el color cuando se presionan cada uno de los 27 botones, esta propiedad le da realismo a la experiencia cuando se está interactuando con los botones.

<table>
<tr><td colspan="3">Escenario: Práctica.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual y replicación de las señas por medio de las imágenes de las señas punteadas.</td></tr>
<tr><td colspan="3">Nombre: Hojas con señas punteadas.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td>  </td><td>Las hojas con las señas punteadas son 3D Object > UI > Canvas > Image de Unity, esté se editó agregando una hoja de cuaderno como imagen de fondo y las 27 imágenes de las señas LSC punteadas diseñadas en Inskape.</td><td>Desaparece.cs

Este script permite a la imagen activar y desactivarse en un tiempo determinado en el escenario de Práctica.</td></tr>
<tr><td colspan="3">Escenario: Prueba.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual e insertar el cubo en la base correspondiente a la palabra que el usuario debe completar.</td></tr>
<tr><td colspan="3">Nombre: 27 cubos del alfabeto.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td>  </td><td>Los 27 cubos son 3D Object>Cube de Unity, a cada uno de los 27 cubos se les añadió las letras del alfabeto en cada una de las caras, acompañadas de diferentes colores.</td><td>InteractionBehavior.cs

Este script facilita la interacción con los objetos, ya que permite sujetar y lanzar los objetos del escenario. En este caso el usuario puede sujetar el cubo con sus manos en la realidad virtual.</td></tr>
</table>

<table>
<tr><td colspan="3">Escenario: Prueba.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad y completar la palabra.</td></tr>
<tr><td colspan="3">Nombre: Bases de las palabras.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td>  </td><td>Las 27 bases corresponden a las palabras incompletas cada una son 3D Object>Cube de Unity, el cual se editó, para que tuviera la forma de un rectángulo, a estas bases se le agrego el material shader Fx/Glass, que en el modo juego le da una transparencia, también a cada una de las bases se le agregaron 27 palabras incompletas con el objetivo de que el usuario las complete con los cubos que se encuentran distribuidos en el escenario prueba.</td><td>AnchorableBehavior.cs

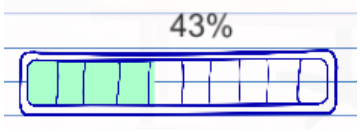
Este script facilita la interacción entre las bases palabras y los cubos del alfabeto. Cabe resaltar que este script se modificó, para que se active una hoja que se llama retroalimentación, mientras el cubo se encuentre dentro del prima.</td></tr>
<tr><td colspan="3">Escenario: Prueba.</td></tr>
<tr><td colspan="3">Actividad: Exploración del escenario en la Realidad Virtual, evaluación y retroalimentación de la seña realizada por el usuario.</td></tr>
<tr><td colspan="3">Nombre: Barra de Retroalimentación.</td></tr>
<tr><td>Objetos</td><td>Descripción</td><td>Script</td></tr>
<tr><td> 

Ilustración 3. Fuente: Elaboración Propia.</td><td>Las hojas con las señas punteadas son 3D Object > UI > Canvas > Image de Unity, esté se editó agregando una hoja de cuaderno como imagen de fondo y una barra diseñadas en Inskape.</td><td>A.cs

Esta barra y el porcentaje funcionan con los datos escritos por MatLab, con el fin de calificar las señas realizadas por el usuario. La barra y porcentaje varían entre:(0 – 100%), Siendo 0%desacierto y 100% el acierto de la seña en la LSC.</td></tr>
</table>

Capítulo IV. Resultados

En este capítulo se presenta una descripción del producto final del proyecto, que cuenta con la composición de cada escenario, el surgimiento del nombre y logo de la aplicación Leap Sign^{VR}, y por último una guía de instrucciones, que debe conocer el usuario antes de usar la aplicación, donde se abarca paso a paso el funcionamiento y las actividades de cada escenario comenzando con el Menú, el escenario de Práctica y el escenario de Prueba.

4.1 Descripción de los Escenarios.

4.1.1 GameObject: Escenario Principal

El escenario principal tiene una estructura física cúbica, contiene objetos estáticos y dinámicos que sirven para crear un ambiente de salón de clase para la experiencia del usuario en la realidad virtual. Los objetos estáticos son aquellos que hacen parte del escenario y no tiene ninguna acción o animación. Entre los objetos estáticos que hacen parte del salón de

clase están: el tablero, escritorio, pupitres, maletas, libros, estantes de libros, afiches entre otros, como se presenta en la figura 25.

Figura 25. Escenario principal de la aplicación, imagen tomada en 360°.
Fuente: Elaboración Propia.

4.1.2 Nombre y logo de la aplicación.

El nombre del proyecto es Leap Sign^{VR}, se debe al uso del Leap Motion, la realidad virtual y al objetivo principal de este trabajo, el cual es el apoyo a la enseñanza de la lengua de señas colombiana; por lo anterior se usa la palabra *Leap* por el nombre del sensor Leap Motion empleado en el proyecto, la palabra *Sign* -escrita en inglés- cuyo significado en español es seña y la sigla *VR* debido a que este proyecto se desarrolló en un ambiente de realidad virtual.

El logo se diseñó en el programa *Inkscape* tomando como referencia el deletreo del nombre de la aplicación (Leap Sign) en la LSC, las cuales fueron capturadas por medio de fotos de la mano derecha, a continuación, se editaron en *Inkscape* las señas y se les dio un aspecto de caricatura, para finalizar se agregaron las letras del nombre de la aplicación, como se presenta en la Figura 26.

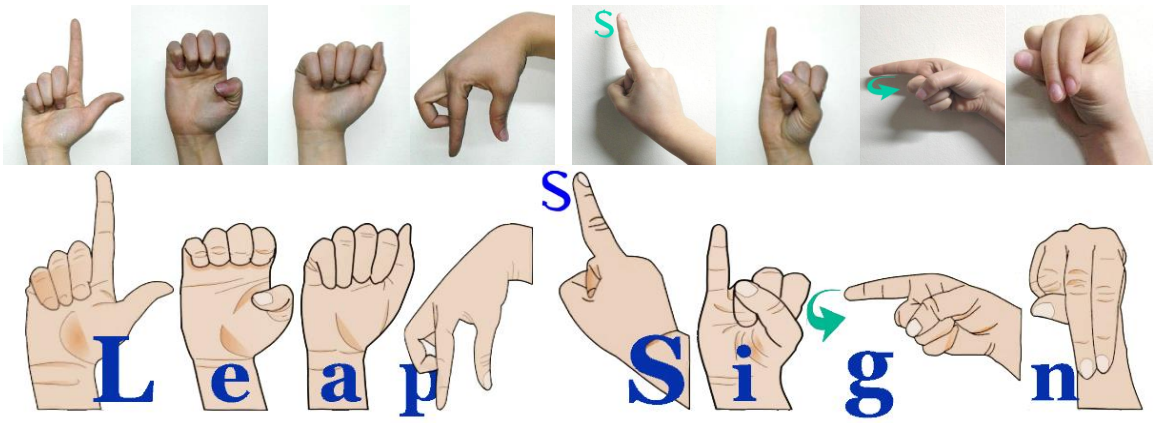


Figura 26. Fotos deletreando el Nombre y logo de la aplicación diseñado en Inkscape.
Fuente: Elaboración propia.

4.1.3 GameObject: Escenario de Menú

El menú tiene una estructura física ovalada, en el interior el usuario se encontrará con el nombre del proyecto Leap Sign[VR] además de una imagen con una breve explicación de cuál es el objetivo de la aplicación, también este escenario tiene Objetos Dinámicos: estos son aquellos objetos que tienen acción, animación y con los que el usuario puede interactuar. Los objetos dinámicos del escenario son: dos botones *UI* que tienen la imagen de reproducir, dos *plane* que contienen los videos de instrucciones de los escenarios:

Práctica y Prueba, dos botones *cube button* que tiene las imágenes con los nombres Práctica y Prueba, dos *Plane* que contiene las instrucciones de forma escrita de Práctica y Prueba, como se presenta en la Figura 27.

Figura 27. Escenario Menú, presentando los videos de Práctica y Prueba.
Fuente: Elaboración Propia.

4.1.4 GameObject: Escenario de Práctica

El escenario de práctica está compuesto de objetos dinámicos: 27 *cube button* con las imágenes de las letras del alfabeto, estos botones se encuentran distribuidos de forma circular en el escenario principal, 27 *plane* con videos de las señas de las letras de la LSC, las señas están acompañadas de su respectiva letra, y por último se encuentran las hojas de cuaderno, las cuales contienen las imágenes punteadas del alfabeto en señas del LSC, como se presenta en la Figura 28.

Figura 28. Escenario Práctica, imagen tomada en 360°.
Fuente: Elaboración Propia.

4.1.5 GameObject: Escenario de Prueba

Este escenario tiene la estructura física del escenario principal, y todos los objetos están distribuidos en el escenario principal. Entre los objetos se encuentran los objetos dinámicos y estos son: 27 cubos 3D con las letras del alfabeto en sus 6 caras, también se encuentran distribuidas las bases que tiene unas palabras y un prisma con las que el usuario interactúa, Figura 29.

Figura 29. Escenario Prueba, imagen tomada en 360°.
Fuente: Elaboración Propia.

4.2 Mecánicas del juego

En esta parte del trabajo se explicarán todas las funcionalidades, a las que se enfrentará el usuario en la experiencia de RV, la cual tiene como objetivo el apoyo y la retroalimentación a la enseñanza del alfabeto de la lengua de señas colombiana.

Al iniciarse la aplicación, la primera pantalla que el usuario observará es una *Animation* que contiene las imágenes: sensor LeapMotion, logo y nombre de la aplicación, esto con la intención de que el usuario se lleve una idea de lo que va a encontrar en la experiencia de RV, Figura 30.

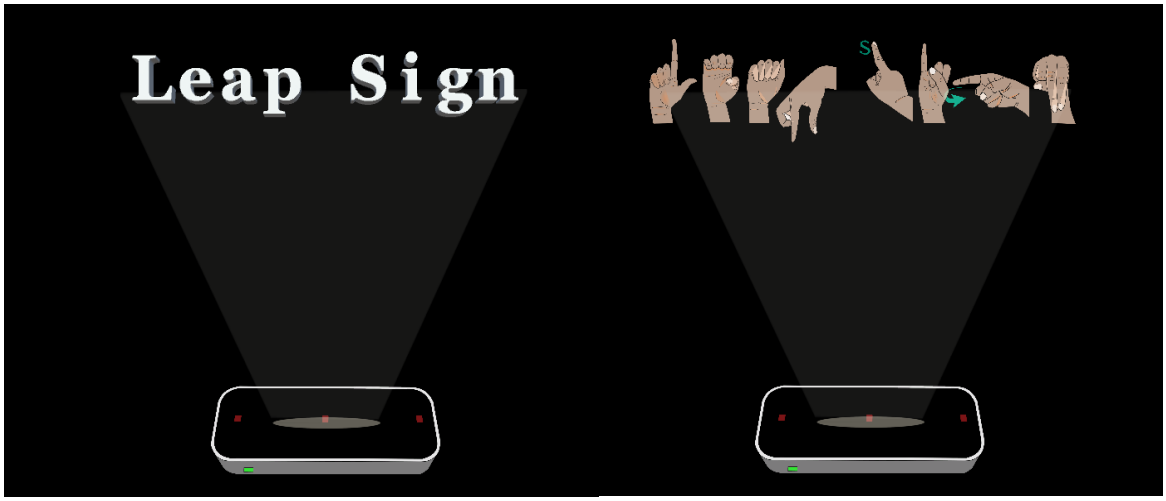


Figura 30. Introducción de la aplicación.
Fuente: Elaboración Propia.

Después de pasar unos segundos, el usuario se encontrará con la primera actividad, la cual está inmersa en todos los escenarios y consiste en experimentar, aprehender y observar en el entorno sobre la LSC, luego el usuario aparecerá en el *escenario menú*, en éste hay una imagen prefijada, que contiene el nombre y una breve explicación de la aplicación, Figura 31.

Figura 31. Inicio de la Primera Actividad. Encuentro del usuario en la Realidad Virtual.
Fuente: Elaboración Propia.

Como es una aplicación en RV el usuario podrá caminar por el escenario menú, éste a su vez, debe presionar el botón azul que tiene el símbolo de reproducir para activar el video de instrucción, y al finalizar el video se activa un botón con el nombre práctica, éste debe ser presionado para acceder a un nuevo escenario “Práctica”, como se presenta en la Figura 32.

Figura 32. Menú, botones para observar las instrucciones de práctica y Prueba en videos.
Fuente: Elaboración Propia.

En el escenario Práctica, iniciará la segunda actividad, allí el usuario estará rodeado por un circulo de 27 botones, cada botón tiene la imagen de la letra del alfabeto, Figura 33. Tan pronto el usuario presione el botón de la letra que quiera conocer en la LSC, se activará un video explicando cómo se debe realizar la seña, Figura 34, enseguida aparece una hoja de cuaderno con la seña punteada, para que el usuario la replique con las manos virtuales, Figura 35.

Figura 33. Escenario Práctica. Antes de presionar el botón.
Fuente: Elaboración Propia.

Figura 34. Escenario Práctica. En el momento de presionar el botón de la letra A.
Fuente: Elaboración Propia.

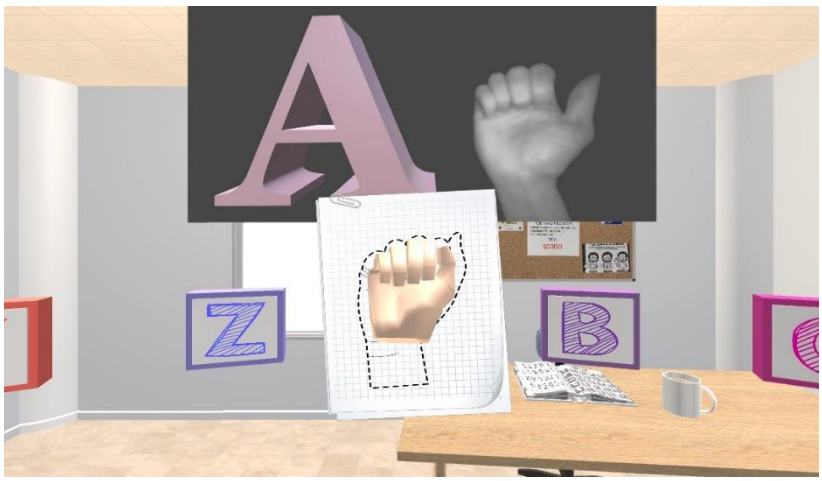


Figura 35. Segunda Actividad: El usuario practicando lo que se observa en el video. Fuente: Elaboración Propia.

En caso en que el usuario haya finalizado las señas del alfabeto y desee salir del escenario práctica, este debe mirar y girar la mano izquierda para que le aparezcan dos botones, uno verde que es para ir al escenario menú y el botón rojo para salir de la aplicación, como se presenta en la figura 36. Cuando el usuario se encuentre en el escenario menú podrá presionar el segundo botón azul que tiene el símbolo de reproducir y se activará el video de instrucción de la actividad prueba y evaluación, cuando éste finalice se activará un botón con el nombre prueba, el usuario debe presionar el botón para acceder a las actividades de prueba y evaluación, Figura 37.

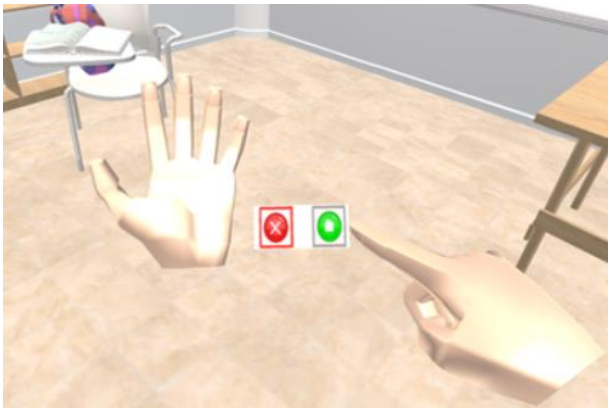


Figura 36. Despliegue de botones de la mano izquierda.
Fuente: Elaboración Propia.

Figura 37. Retornando al Menú para que el usuario se dirija al escenario de prueba.
Fuente: Elaboración Propia.

El usuario se encuentra en la tercera actividad llamada prueba, aquí el usuario se encontrará con cubos que tienen las letras del alfabeto y con un grupo de palabras incompletas, que se encuentran distribuidos por todo el escenario, con la intención de que el usuario complete las palabras con el cubo correspondiente a la letra faltante, Figura 38.

Para empezar el usuario debe agarrar el cubo e insertarlo en un prisma que se encuentra al lado de la palabra incompleta, como consecuencia de la unión de estos dos objetos se activará la cuarta y última actividad llamada evaluación, que consiste en una hoja de cuaderno, que tiene como fin evaluar la seña que deberá hacer el usuario de acuerdo con la letra que le aparecerá en la hoja.

Figura 38. Tercera Actividad. Insertar cubo en la palabra incompleta.
Fuente: Elaboración Propia.

La cuarta y última actividad se encuentra dividida en tres momentos: primer momento se activará una hoja (en la parte superior aparecerá la letra del cubo), en la mitad de la hoja la palabra *Listo* (le da un tiempo al usuario para que este frente a la hoja), y en la parte inferior de la hoja una *barra* (que estará llenándose o vaciándose si la seña no está bien realizada), como se presenta en la Figura 39.

Figura 39. Cuarta Actividad. Primer momento, hoja de retroalimentación.
Fuente: Elaboración Propia.

En el segundo momento aparece un *contador* en la parte superior derecha de la hoja este es el tiempo en que el usuario debe realizar las señas, Figura 40, y en el tercer momento se activarán unas imágenes para calificar las señas, estas serán calificadas con las imágenes de: *bien o mal*. Figura 41. Cabe resaltar, que las señas sin movimiento son evaluadas de acuerdo al entrenamiento realizado en las redes neuronales, y las señas con movimiento (J, S y Z), se diseñaron con unos puntos, los cuales el usuario activa a medida que hace contacto con la mano virtual, esto con el fin de generar la trayectoria que se realiza en la dactilología de la LSC.

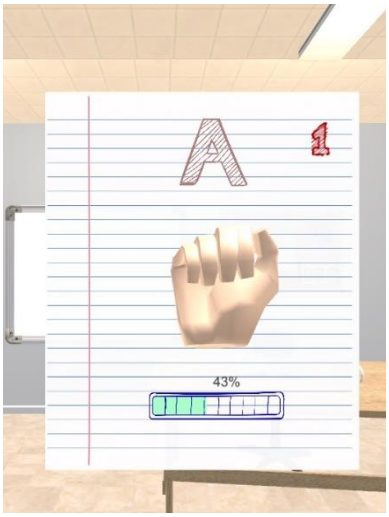


Figura 40. Segundo momento. Evaluación y retroalimentación de la seña A en tiempo real. Fuente: Elaboración propia.

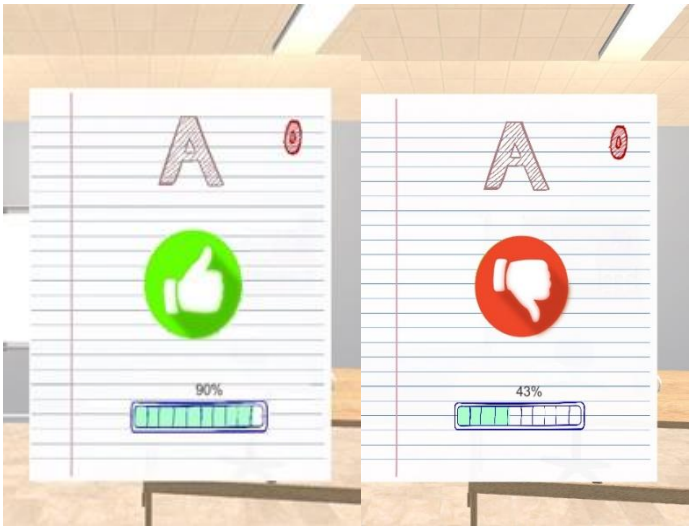


Figura 41. Tercer momento. Resultado de la evaluación de la seña A, izq. Ejecución correcta, der. Ejecución incorrecta. Fuente: Elaboración Propia.

Finalmente, la aplicación le mostrará al usuario el resultado de la evaluación y saldrá de allí automáticamente, luego el usuario tendrá la posibilidad de seguir siendo evaluado con las otras letras o ir al menú principal a través del botón menú, para así salir de la aplicación a través de un botón de salir.

Capítulo V. Conclusiones

A continuación, se exponen una serie de resultados que muestran tanto el proceso como las dificultades que se presentaron durante el desarrollo del proyecto. A modo de recomendaciones se formularán pautas para la continuación de este.

En primera medida, Leap Sign^{VR} se desarrolló con el fin de generar una herramienta que apoye a la enseñanza del alfabeto de la LSC, y permita la inclusión de la comunidad sorda en un entorno educativo con ayuda de la Realidad Virtual y el sensor Leap Motion.

Con el fin de lograr lo anteriormente expuesto, se diseñó un salón de clase porque brinda un entorno pedagógico similar a un salón real.

Por otra parte, se seleccionó el alfabeto de la LSC, porque el conjunto de consonantes y vocales le permite a la comunidad sorda poder comunicarse a través del deletreo, además como es una aplicación educativa brinda la posibilidad de que las personas oyentes aprendan esta lengua de signos.

El diseño de las cuatro actividades: exploración de los escenarios, observación de los vídeos, completar las palabras y la evaluación de las señas, fueron resultado de una serie de investigaciones, donde se observaron diferentes herramientas que se enfocan en el aprendizaje unidireccional, es decir, una comunicación entre el humano-máquina por medio de imágenes y videos que no permite una retroalimentación por parte del usuario, por este motivo se desarrollaron actividades en Realidad Virtual que permiten un aprendizaje bidireccional, es decir, el usuario puede interactuar, aprender, reforzar y retroalimentar el alfabeto de LSC en tiempo real.

El desarrollo de los algoritmos en el proyecto es un punto que resaltar, porque con ello se logró la retroalimentación y evaluación del alfabeto de la LSC, a través de la captura de las posiciones cartesianas espaciales de la mano derecha, e identificando cada seña por medio del algoritmo de la resta de la posición de la palma con cada uno de los dedos, con el fin de que la seña no se distorsione de acuerdo con la posición en el espacio.

Se concluyó que el sensor Leap Motion está limitado a la hora de detectar la posición de los dedos en la RV, debido a que el sensor está ubicado sobre las gafas oculus y el jugador debe realizar las señas a sí mismo, a causa de esto las señas del alfabeto de la LSC : M, N y Ñ no se pueden realizar satisfactoriamente, puesto que el sensor no las diferencia, por la posición de los dedos y el dorso de la mano, además era incomodo realizar estas señas de frente al sensor Leap Motion por consiguiente se recomienda hacer una pequeña modificación en la rotación a las señas mencionadas.

Después de realizar las evaluaciones a los tipos de entrenamientos Levenberg-Marquardt con un acierto del 68.8% y Regularización Bayesiana con un acierto de 83.7%, al momento de realizar el alfabeto de LSC, se concluyó que el mejor tipo de entrenamiento para este caso es la Regularización Bayesiana.

Como trabajo a futuro, Leap Sign^{VR} puede ser complementado con el uso de palabras cotidianas de la lengua de señas colombiana, además para una mayor precisión en la realización de las señas se recomienda trabajar con dos sensores Leap Motion.

Para finalizar, Leap Sign^{VR} es una aplicación que está enfocada en la parte educativa para apoyar la enseñanza del alfabeto de LSC, con el fin de reducir las brechas de la comunicación entre la comunidad sorda y la oyente, y por parte de la tecnología Leap Sign^{VR} está enfocada en las experiencias que genera el entorno de la Realidad Virtual beneficiando el uso y la interacción con los usuarios.

Capítulo VI. Bibliografía

Aggarwal, R., Swetha, S., Namboodiri, A. M., Sivaswamy, J., & Jawahar, C. V. (2015). Online handwriting recognition using depth sensors. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, (págs. 1061 - 1065). Tunis.

Bairathi , D., & Gopalani, D. (2017). Opposition-Based Sine Cosine Algorithm (OSCA) for Training Feed-Forward Neural Networks. *13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, (págs. 438-444). Jaipur.

Bernal Villamarin, S. C., Cantor Morales, D. A., Ávila Reyes, C. A., & Sánchez, C. A. (2016). Application design sign language colombian for mobile devices VLSCApp (Voice Colombian sign language app) 1.0. *2016 Technologies Applied to Electronics Teaching (TAEE)*, (págs. 1-5). Sevilla.

Chuan, C. H., Regina, E., & Guardino, C. (2014). American Sign Language Recognition Using Leap Motion Sensor. *13th International Conference on Machine Learning and Applications*, (págs. 541 - 544). Detroit.

Demircioğlu, B., Bülbül, G., & Köse, H. (2016). Turkish Sign Language recognition with Leap Motion. *Signal Processing and Communication Application Conference (SIU)*, (págs. 589 -592). Zonguldak.

Ebrahim Al-Ahdal, M., & Nooritawati, M. T. (2012). Review in Sign Language Recognition Systems. *Symposium on Computers & Informatics (ISCI)*, (págs. 52 - 57). Penang.

Freeman, I., Salmon, J., & Coburn, J. (2016). CAD Integration in Virtual Reality Design Reviews for Improved Engineering Model Interaction. *ASME 2016 International Mechanical Engineering Congress and Exposition*, (pág. V011T15A006). Phoenix, Arizona.

Garcia, M. G., Luis , C. I., & Samonte, M. J. (2016). E-tutor for Filipino Sign Language. *2016 11th International Conference on Computer Science & Education (ICCSE)*, (págs. 223-227). Nagoya.

Geetha, M., Manjusha, C., Unnikrishnan, P., & Harikrishnan, R. (s.f.). A vision based dynamic gesture recognition of Indian Sign Language on Kinect based depth images. *2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications, IEEE-C2SPCA 2013*, (pág. 2013).

Herrera, V. (2005). Adquisición Temprana de Lenguaje de Signos y Dactilología. *Revista Psicopedagógica*, 2-10.

Inkscape. (2018). https://inkscape.org/es/ .

INSOR. (2006). *Diccionario básico de la lengua de señas colombiana.* Bogotá: Instituto Nacional para Sordos e Instituto Caro y Cuervo.

Jeisson Pérez. (2015). Sings2Me. *WiseWare S.A.S*.

Khalil, M. A., & Kotaiah, B. (2017). Implementation of agile methodology based on SCRUM tool. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, (págs. 2351-2357). Chennai.

Ladrón de Guevara, M. Á. (2018). *Técnicas de comunicación con personas dependientes en instituciones.* Logroño: Tutor Formación.

Leap Motion. (2018). *https://www.leapmotion.com/*.

Ling, H., & Rui, L. (2016). VR glasses and leap motion trends in education. *2016 11th International Conference on Computer Science & Education (ICCSE)*, (págs. 917-920). Nagoya.

Liou, W.-K., & Chang, C.-Y. (2018). Virtual reality classroom applied to science education. *23rd International Scientific-Professional Conference on Information Technology (IT)*, (págs. 1 - 4). Zabljak.

MathWorks. (2018). https://www.mathworks.com/products/matlab.html.

Medjram, S., Babahenini, M. C., Taleb-Ahmed, A., & Bed Ali, Y. M. (2016). *Real-time wrist localization in color images based on corner analysis.* New York.

Modanwal, G., & Sarawadekar, K. (2018). *A RobustWrist Point Detection Algorithm using Geometric Features.* Varanasi.

Mohandes, M., Aliyu, A. S., & Deriche, M. (2014). Arabic sign language recognition using the leap motion controller. *23rd International Symposium on Industrial Electronics (ISIE)*, (págs. 960 - 965). Istanbul.

Motion, L. (2018). https://www.leapmotion.com/.

Oculus. (2018). https://www.oculus.com/.

Orza, J. G. (2002). Neuropsicología cognitiva de la lengua de signos: una piedra de toque para el estudio del lenguaje, la visión, las emociones y el movimiento. *: Revista de psicología general y aplicada: Revista de la Federación Española de Asociaciones de Psicología*, 89-104.

Parra, C. (2010). Educación inclusiva : Un modelo de educación para todos. *Revista Isees*, 73-84.

Paulraj, M. P., Yaacob, S., bin Zanar Azalan, M. S., & Palaniappan, R. (2010). A phoneme based sign language recognition system using skin color segmentation. *2010 6th International Colloquium on Signal Processing & its Applications*, (págs. 1 - 5). Mallaca City.

Pertusa Venteo, E., & Fernandez Viader, M. D. (2005). *El valor de la mirada: sordera y educación.* Barcelona.

Ravikiran, J., Mahesh, K., Mahishi, S., Dheeraj, R., Sudheender, S., & Nitin, V. (2009). Finger Detection for Sign Language Recognition. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I. IMECS 2009.* Hong Kong.

Ren, Z., Yuan, J., & Zhang, Z. (2014). Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *2011 Proceedings of the 19th ACM international conference on Multimedia - MM '11*.

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *2017 International Conference on Computing, Communication and Automation (ICCCA)*, (págs. 864-869). Greater Noida.

Steinicke, F. (2016). *Being Really Virtual Immersive Natives and the Future of Virtual Reality*. Cham: Springer.

SubVRsive. (2018). *https://subvrsive.com/announcing-project-asl-american-sign-language-mixed-reality/*.

Tablada, C. J., & Torres, G. A. (2015). Redes Neuronales Artificiales. *Revista de Educación Matemática*, 22-30.

TechSmith. (2018). https://www.techsmith.com/video-editor.html.

Thepade, S. D., Kulkarni, G., Narkhede, A., Kelvekar, P., & Tathe, S. (2013). Sign language recognition using color means of gradient slope magnitude edge images. *2013 International Conference on Intelligent Systems and Signal Processing (ISSP)*, 216 - 220.

Unity Technologies. (2018). *https://unity3d.com/es*.

Vince, J. (2011). *Introduction to Virtual Reality.* Tunbridge Wells: Springer.

Wibowo, M. D., Nurtanio, I., & Ahmad Ilham, A. (2017). Indonesian sign language recognition using leap motion controller. *11th International Conference on Information & Communication Technology and System (ICTS)*, (págs. 67 -72). Surabaya.

Xu, Y., Gu, J., & Tao, Z. (2009). Bare Hand Gesture Recognition with a Single Color Camera. *2nd International Congress on Image and Signal Processing*, (págs. 1 - 4). Tianjin.

Yeh, W. Y., Tseng, T. H., Hsieh, J. W., & Tsai, C. M. (2016). Sign language recognition system via Kinect: Number and english alphabet. *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*, (págs. 660 - 665). Jeju.