Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de computación

Interacción aumentada con objetos articulados

Tesis que presenta

Rosaura Hernández García

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. Luis Gerardo de la Fraga

Ciudad de México Noviembre de 2021

Resumen

La realidad aumentada combina elementos virtuales con escenas físicas reales, con la finalidad de facilitar el entendimiento de objetos o fenómenos poco usuales en entornos reales de la vida cotidiana. De esta forma se podría visualizar el campo magnético, por ejemplo. La visualización del mundo aumentado es sobre un monitor en el que visualizan objetos virtuales sobre una imagen del mundo real. La visualización es en tiempo real a 30 imágenes por segundo.

En esta tesis se construyó un sistema de realidad aumentada que permite la interacción con objetos virtuales articulados animados. El hardware del sistema consiste en una serie de tres marcadores basados en tipo de orden, un sensor de posición para fijar la distancia entre la punta del sensor y los marcadores, una cámara de una Playstation 3, un monitor y la computadora. El software de la aplicación se construyó con la biblioteca de desarrollo de interfaces gráficas Qt y en C++. Con el marcador se calcula la posición y perspectiva de él para posteriormente dibujar el objeto virtual sobre él. Se diseñaron y construyeron tres objetos virtuales, usando el software Blender: una trucha, un pez animado y una grulla. Se usó el modelo matemático de la ecuación de onda para animar el movimiento ondulatorio del nado de los peces y del vuelo del ave. Para poder animar los objetos virtuales se definen estados y comportamientos; estos comportamientos son secuencias de estados. La animación se implementa en software como máquinas de estados.

Se realiza una detección de colisiones entre la punta del sensor de posición y el objeto virtual y si existe colisión se ejecuta un comportamiento del objeto virtual. La detección de colisiones permite realizar la interacción con los peces y el ave, por ejemplo, el comportamiento de huida se realiza cuando se acerca el sensor a alguno de los objetos virtuales.

Abstract

Augmented Reality combines virtual elements within real physical scenes, with the idea to aim the understanding of objects or non-usual phenomena within real life environments. Using this technology is possible to visualize a magnetic field, for example. The augmented world is watched on a monitor screen in which the virtual objects are projected over an real world image. Visualization is performed in real time at 30 images per second.

In this Thesis, an Augmented Reality system has been built. Furthermore, this system allows the human interaction with animated and articulated virtual objects. The system hardware consist in three fiducial markers based in order type, one position sensor to know the interaction position, one Playstation 3 camera, a display and a computer. Application software has been built with the development library for graphical user interfaces Qt and in C++. A marker allows to calculate its position and perspective transformation to draw above it a virtual object. Three virtual objects are designed and built using Blender program: one trout, one animated fish, and one crane. To animate these virtual animals, the wave equation is used. Wave equation can modeled the undulatory movement of the fished, and the bird flight. To animate an animal, states are defined, and a sequence of states produce an behavior. This animation is implemented in software are state machines.

A collision detection allows to perform the iteration with the virtual objects. If a collision exists, then a behavior of the virtual object is executed. Collision detection allow the interaction with the fishes and the bird, for example, the behavior to run away is performed when the sensor is bring closer to an animal.

Agradecimientos

A CONACYT por haberme dado el apoyo económico para estudiar la maestría.

Al CINVESTAV y al Departamento de Computación por aceptarme en su programa de estudios y brindarme un lugar y ámbiente de trabajo bonito mientras se pudo.

Al Dr. Luis Gerardo de la Fraga, quien me asesoró y guió para poder realizar esta tesis, y de quién aprendí no sólo de conocimientos técnicos, también lecciones de vida.

A la Dra. Brisbane por darme el consejo de cursar las materias que más me interesaran, aún cuando tuviera miedo de la dificultad en ellas, así llegué hasta esta tesis. Al Dr. Cuauhtemoc por su paciencia y dedicación al impartir sus clases, me dió confianza para seguir. A ambos por ayudar a la revisión de este documento.

A mi familia por siempre apoyarme, y por ayudarme a conseguir el entorno de trabajo que necesitaba para desarrollar este proyecto. Especialmente a mi papá por siempre enseñarnos que se puede lograr todo con dedicación, a mi mamá por enseñarnos a ser pacientes, a mi hermano por ser una inspiración para dar lo mejor de mí y a mi sobrina por supervisar mi trabajo.

A mi amigo Ricardo quien siempre me da el mejor de los consejos.

Índice general

Índice de figuras

Capítulo 1

Introducción

La realidad aumentada (en adelante RA) es el término que se usa para describir un sistema que realiza la integración de objetos virtuales en imágenes reales, permitiendo al usuario percibir un entorno diferente al que está acostumbrado sin alejarse del todo de la realidad. A diferencia de la realidad virtual (en adelante RV), la cual presenta al usuario un escenario totalmente sintético [1].

Actualmente, existen sistemas de realidad aumentada dedicados al entrenamiento en las áreas de danza, deporte, educación y desde el punto de vista médico [2]. Estos sistemas son de gran importancia para prácticas que pueden ser difíciles de reproducir en el plano real.

En esta tesis se quiere resolver el problema de la interacción con objetos articulados, los cuales puedan reaccionar demostrando un comportamiento real ante la interacción del usuario. Este tipo de soluciones son pensadas para poder utilizarse a futuro en distintas áreas educativas.

Desde hace ya varios años es cada vez más importante y útil el uso de tecnologías como la realidad aumentada para las técnicas de aprendizaje. Particularmente, la realidad aumentada se ha utilizado en ámbitos como la medicina, ya que permite realizar prácticas que en la vida real son costosas y en ocasiones riesgosas. Para poder realizar estas interacciones los sistemas de realidad aumentada, utilizan distintos tipos de sensores e interfaces hápticas [3]. Las interfaces hápticas permiten tocar, sentir o manipular objetos simulados en un escenario virtual, con lo cual el usuario experimenta una sensación de inmersión dentro del espacio virtual además de establecer entre el usuario y el entorno virtual una trasnferencia bidireccional de información en tiempo real [4, 5, 6]. En los últimos años ha habido un gran avance en el uso de interfaces hápticas para el entrenamiento médico [3, 2].

1.1. Planteamiento del problema

En esta tesis se desarrollará un sistema de realidad aumentada con objetos articulados en el que se podrá interactuar con éstos a través de un sensor que nos entregue la posición de la mano. Aquí se usará una interfaz háptica solo como un sensor para tener la posición xyz de la mano. El sistema reconocerá un par de marcadores mostrando un ave o un pez dependiendo de cada marcador y el usuario será capaz de hacer la interacción con éstos. El ave o el pez serán objetos virtuales dinámicos que tendrán movimiento propio y reaccionarán a la interacción con la mano.

En la figura 1.1 podemos observar el diagrama general de un sistema de realidad aumentada: se captura un marco de video, que es una imagen que se obtiene a través de una cámara. Luego, la imagen se procesa con técnicas de procesamiento de imagen para poder detectar el marcador y se obtiene la posición y orientación de este. Con la información de la orientación y posición del marcador se puede dibujar el objeto virtual respecto a esa posición y orientación. Se tiene también la entrada del sensor, con la cual se realiza la interacción con el objeto virtual. Además, en el diagrama se resaltan las partes en las que se enfoca este trabajo, las cuales son dibujar el objeto virtual articulado y la interacción con éste.

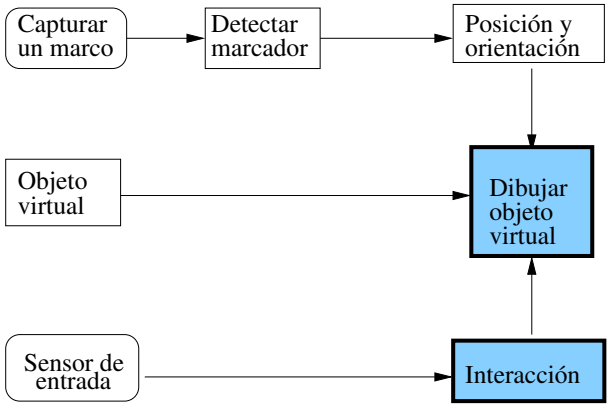


Figura 1.1: Diagrama general del sistema de realidad aumentada.

En la figura 1.2 se muestra un diagrama específico del sistema donde se presentan sus componentes: la cámara, el sensor ultrasónico, la computadora, la pantalla (arriba a la derecha, es otro monitor) donde se mostrará el objeto virtual. En el diagrama, se observa que es en el monitor donde se tiene la interacción que hará el usuario con el objeto virtual. En el mundo virtual la posición y orientación del sensor se puede ver como un solo dedo. El funcionamiento de los componentes en la figura 1.2 es el siguiente: arriba del marcador en el mundo virtual se dibuja el pez que estará en movimiento. Al acercar el dedo, el pez reaccionará y huirá de donde está la posición del dedo. Al quitar el dedo, el pez regresará nadando a su posición inicial, donde seguirá aleteando. El pez también puede moverse moviendo el marcador.

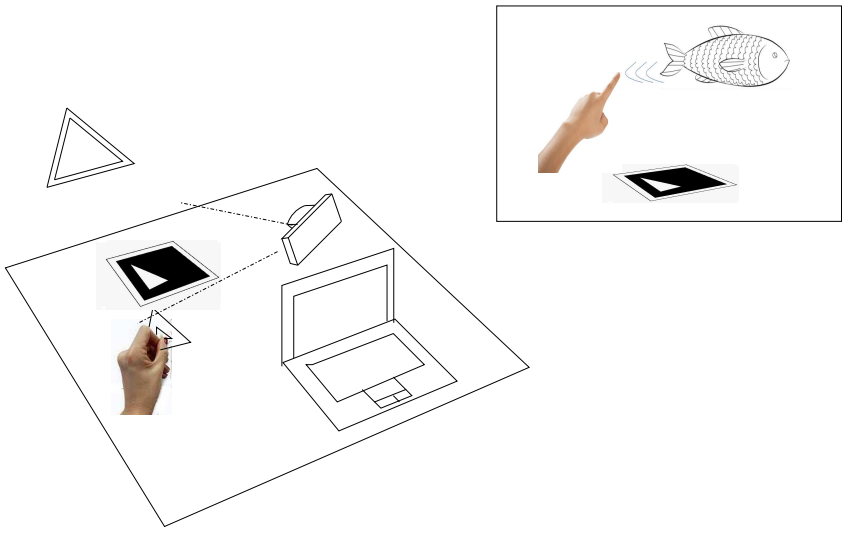


Figura 1.2: Diagrama específico del sistema de realidad aumentada con objetos articulados.

1.2. Objetivos

1.2.1. General

El desarrollo de un sistema de realidad aumentada con objetos articulados, siendo éstos un ave y un pez con los cuales se podrá interactuar a través de un sensor de posición.

1.2.2. Particulares

1. Adicionar el módulo de seguimiento del marcador al sistema de realidad virtual que actualmente se tiene, el cual se desarrolló durante el curso de Visión del cuatrimestre Mayo-Agosto 2020 en el CINVESTAV y que contiene el reconocimiento de un marcador de cinco puntos y la visualización de un tetraedro.

2. Agregar el reconocimiento de varios marcadores.

3. Integrar el sensor o interfaz háptica al sistema para tener la posición de la mano.

4. Estudio sobre los objetos articulados.

5. Diseñar el pez y el ave que se usarán.

6. Implementar simulaciones del movimiento de acuerdo a descripciones físicas de cada uno de los objetos virtuales, esto es para un pez y un ave.

7. Integrar todo el sistema.

1.3. Organización de la tesis

La presente tesis está organizada en 5 capítulos:

El presente capítulo donde como introducción se dió la definición de RA y su diferencia con la RV. Se expuso el plateamiento del problema, así como los objetivos específicos y particulares a resolver en esta tesis.

En el capítulo 2 se muestran los conceptos teóricos necesarios para entender el funcionamiento del sistema. Tales como las transformaciones geométricas 3D, las cuales son escenciales dentro de la gráficación por computadora y por tanto de la RA. El modelo de la cámara obscura, que representa la relación entre una escena 3D y una fotografía tomada por la cámara. El cálculo de una homografía, la calibración de la cámara una partir de homografías, marcadores basados en tipo de orden detección del marcador y detección de colisiones.

Dentro del capítulo 3 se presenta el sistema de Realidad Aumentada. Durante el curso de Visión impartido en el periodo Mayo-Agosto de 2020 se desarrolló una aplicación de realidad aumentada. Es esta aplicación a la que se le agregó el módulo de seguimiento del marcador, ya que el reconocimiento se encontraba dentro de un bucle que continuaba buscando el marcador a pesar de haberlo encontrado una primera vez, por lo que en ocasiones se perdía la continuidad del objeto virtual.

En el capítulo 4 se presenta la metodología del desarrollo y validación del sistema.

En el capítulo 5 se dan las conclusiones y el trabajo a futuro.

Capítulo 2

Marco teórico

En el presente capítulo se exponen los fundamentos teorícos más importantes para el desarrollo de esta tesis.

2.1. Transformaciones geométricas en 3D

En el ámbito de las gráficas por computadora, las transformaciones geométricas constituyen un punto clave pues son la base para construir escenarios tridimensionales [7]. De forma puntual, la traslación, rotación y escalamiento son indispensables para esta tarea, pues es mediante la composición de transformaciones que se consigue posicionar los objetos en una determinada región del espacio.

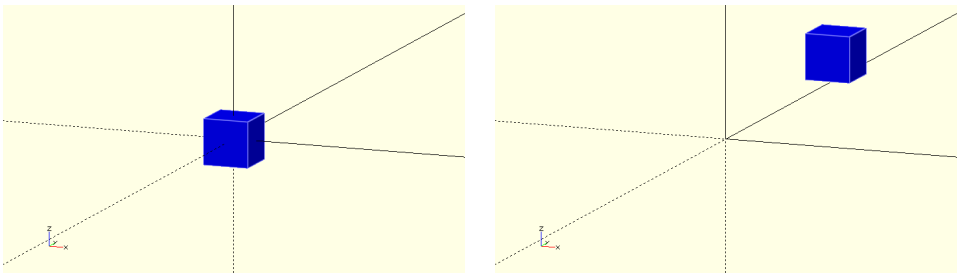
2.1.1. Traslación

La traslación es la transformación que permite cambiar la posición de un objeto desplazándolo hacia un punto en el espacio. Las coordenadas se obtienen mediante las siguientes ecuaciones:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + z_x \end{aligned} \tag{2.1}$$

donde t_x , t_y, t_z son los valores que se translada el punto $[x, y, z]^{\mathrm{T}}$ para resultar en el punto $[x', y', z']^{\mathrm{T}}$.

Como vemos todos los puntos serán representados como vectores columna. Otra forma de representar la traslación, es en su forma matricial a través de puntos tridimensionales homogéneos:



(a) Objeto original. (b) Objeto trasladado.

Figura 2.1: Ejemplo de traslación.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

La ecuación (2.2) la podemos simplificar como:

$$\boldsymbol{P'} = T(t_x, t_y, t_z)\boldsymbol{P}, \tag{2.3}$$

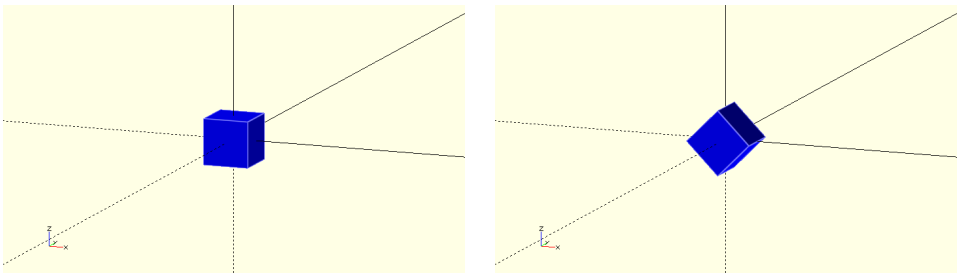
donde T es la matriz de tamaño 4×4 en (2.2) y $\boldsymbol{P'}$ y $\boldsymbol{P}$ son los puntos trasladado y original, respectivamente.

En la figura 2.1 se muestra un ejemplo de traslación de un objeto, podemos observar que éste conserva su forma y tamaño.

2.1.2. Rotación

La rotación es la transformación geométrica que permite cambiar la orientación de los objetos en el espacio. Las rotaciones principales 3D son aquellas que se dan en alguno de los ejes principales, x, y, z.
Por convención, los ángulos de rotación positivos dan como resultado rotaciones en contra de las manecillas del reloj sobre el eje de rotación [8].
Las ecuaciones de rotación, en su forma homogénea, para los ejes principales x, y, z, se expresan como:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\operatorname{sen}(\theta) & 0 \\ 0 & \operatorname{sen}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$



(a) Objeto original. (b) Objeto rotado.

Figura 2.2: Ejemplo de rotación.

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \text{sen}(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\text{sen}(\theta) & 0 & 0 \\ \text{sen}(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

donde θ es el ángulo de rotación. En la figura 2.2 se muestra un ejemplo de rotación de un objeto.

2.1.3. Escalamiento

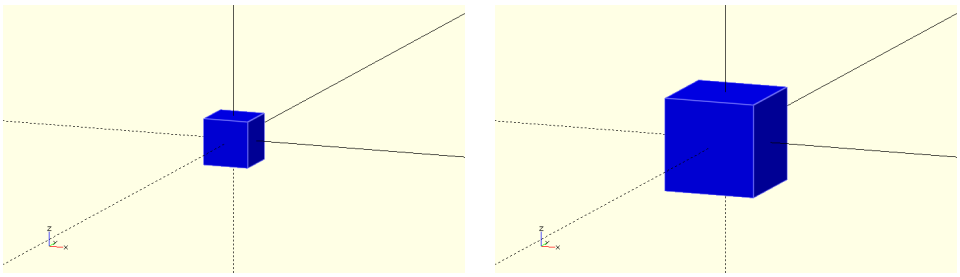
El escalamiento es la transformación geométrica que permite cambiar el tamaño de un objeto. Se aplica una multiplicación sobre todos los puntos que conforman un objeto para que su proporción cambie.
El escalamiento de un punto tridimensional se define como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} e_x & 0 & 0 & t_x \\ 0 & e_y & 0 & t_y \\ 0 & 0 & e_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

En la figura 2.3 se muestra un ejemplo de escalamiento de un objeto.

2.2. Modelo de la cámara oscura

El modelo de la cámara obscura define la relación entre la ubicación de un punto **P**, en una escena del mundo 3D y la proyección de ese punto, a un punto **p**, en una imagen 2D [9]. En la figura 2.4 se muestra esta relación.



(a) Objeto original. (b) Objeto escalado.

Figura 2.3: Ejemplo de escalamiento.

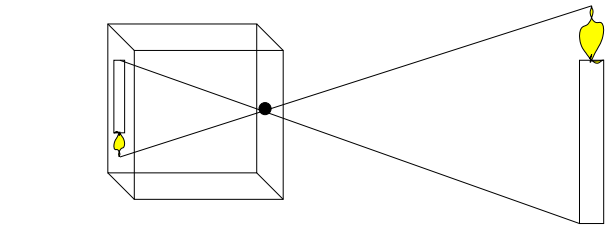


Figura 2.4: Modelo de la cámara oscura.

Dado un punto tridimensional $\mathbf{P}$ cuyas coordenadas homogéneas son $\mathbf{P} = [x, y, z, 1]^T$, y su correspondiente punto en la imagen es $\mathbf{p} = [u, v, 1]^T$. La relación entre un punto $\mathbf{p}$ de la imagen y el punto $\mathbf{P}$ en el mundo 3D es:

$$\lambda \mathbf{p} = A\mathbf{P} \tag{2.8}$$

donde A es una matriz de tamaño 3×4 que contiene la información para rotar, trasladar y proyectar un punto tridimensional $\mathbf{P}$ a un punto $\mathbf{p}$ en una imagen 2D y λ es un factor de escala.

La matriz A en (2.8) se puede definir como:

$$A = KM \tag{2.9}$$

donde M es una matriz aumentada compuesta por los parámetros extrínsecos $[R|\boldsymbol{t}]$, R es una matriz de rotación de tamaño 3×3 y $\boldsymbol{t}$ es un vector de traslación de tamaño 3. K es una matriz de tamaño 3×3 que se conoce como matriz de parámetros intrínsecos y calcula la transformación de proyección en perspectiva.

Al realizar la sustitución de A en la ecuación (2.8) se obtiene:

$$\lambda \mathbf{p} = K\,[R|\boldsymbol{t}]\,\mathbf{P} \tag{2.10}$$

Desarrollando la ecuación (2.10) se obtiene:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.11}$$

El punto 3D $\mathbf{P}$ primero es rotado por R y luego trasladado según el vector $\boldsymbol{t}$. f en la matriz K es la distancia focal, (u_0, v_0) representa el punto principal de la cámara.

La ecuación (2.10), puede modificarse sacando R de la matriz aumentada $[R|t]$, con lo que queda de la siguiente manera:

$$\lambda \mathbf{p} = KR\,[I|-\boldsymbol{c}]\,\mathbf{P} \tag{2.12}$$

donde K es la matriz de la cámara, $\boldsymbol{c} = -R^T\boldsymbol{t}$, es el punto donde se sitúa la cámara. R es una matriz de rotación y se cálcula con las siguientes fórmulas:

$$\boldsymbol{z_c} = \frac{\boldsymbol{c} - \boldsymbol{o}}{\|\boldsymbol{c} - \boldsymbol{o}\|} \tag{2.13}$$

$$\boldsymbol{x_c} = \frac{\boldsymbol{z} \times \boldsymbol{z_c}}{\|\boldsymbol{z} \times \boldsymbol{z_c}\|} \tag{2.14}$$

$$\boldsymbol{y_c} = \boldsymbol{z_c} \times \boldsymbol{x_c} \tag{2.15}$$

$$R = \begin{bmatrix} \boldsymbol{x_c^T} \\ \boldsymbol{y_c^T} \\ \boldsymbol{z_c^T} \end{bmatrix} \tag{2.16}$$

donde $\boldsymbol{z}$ es el vector arriba del plano y $\boldsymbol{o}$ es un punto a donde ve la cámara.

2.3. Cálculo de una homografía

La homografía o transformación de perspectiva [10] determina la correspondencia entre dos planos, de forma que cada uno de los puntos en uno, le corresponden, respectivamente, a un punto del otro plano como se observa en la figura 2.5.

La homografía H, está definida como:

$$\lambda \mathbf{p}_2 = H\mathbf{p}_1 \tag{2.17}$$

donde $\boldsymbol{p}_2$ y $\boldsymbol{p}_1$ son puntos 2D homogéneos y λ es un factor de escala.

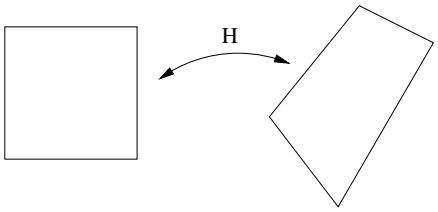


Figura 2.5: Homografía entre dos planos.

De la ecuación (2.10), de la cámara obscura. $\mathbf{P}$ son puntos 3D homogéneos. $\mathbf{p}$ son puntos 2D homogéneos, en el plano de proyección. K es la matriz de parámetros intrínsecos de la cámara. R es una matriz de rotación. $\boldsymbol{t}$ es un vector de tamaño 3×1.

Si tenemos un plano, y suponemos que está situado en el plano $x - y$, y se asume que $\boldsymbol{z} = 0$, (2.10) queda como:

$$\lambda \mathbf{p} = K \begin{bmatrix} r_1 & r_2 & r_3 & | & t \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \tag{2.18}$$

y queda como:

$$\lambda \mathbf{p} = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.19}$$

$$\lambda \mathbf{p} = \lambda \mathbf{p}_2 = M \mathbf{p}_1 \tag{2.20}$$

Entonces M es una homografía y es una matriz de tamaño 3×3. Para obtener la homografía se usa la forma de la ecuación (2.17), está se expande y queda de la siguiente forma:

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{2.21}$$

$$\begin{aligned} \lambda x_2 &= h_{11}x_1 + h_{12}y_1 + h_{13} \\ \lambda y_2 &= h_{21}x_1 + h_{22}y_1 + h_{23} \\ \lambda &= h_{31}x_1 + h_{32}y_1 + h_{33} \end{aligned} \tag{2.22}$$

Se elimina el factor de escala λ, resultando:

$$\begin{aligned} (h_{31}x_1 + h_{32}y_1 + h_{33})x_2 &= h_{11}x_1 + h_{12}y_1 + h_{13} \\ (h_{31}x_1 + h_{32}y_1 + h_{33})y_2 &= h_{21}x_1 + h_{22}y_1 + h_{23} \end{aligned} \tag{2.23}$$

de (2.23) se obtiene:

$$\begin{aligned} h_{31}x_1x_2 + h_{32}y_1x_2 + h_{33}x_2 &= h_{11}x_1 + h_{12}y_1 + h_{13} \\ h_{11}x_1 + h_{12}y_1 + h_{13} - h_{31}x_1x_2 - h_{32}y_1x_2 - h_{33}x_2 &= 0 \end{aligned} \tag{2.24}$$

$$\begin{aligned} (h_{31}x_1 + h_{32}y_1 + h_{33})y_2 &= h_{21}x_1 + h_{22}y_1 + h_{23} \\ h_{31}x_1y_2 + h_{32}y_1y_2 + h_{33}y_2 &= h_{21}x_1 + h_{22}y_1 + h_{23} \\ h_{21}x_1 + h_{22}y_1 + h_{23} - h_{31}x_1y_2 - h_{32}y_1y_2 - h_{33}y_2 &= 0 \end{aligned} \tag{2.25}$$

Para encontrar todos los elementos de H se ubican todos sus elementos en un solo vector $h = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}]$. Y se forma el sistema de ecuaciones:

$$A\mathbf{h} = \mathbf{b} \tag{2.26}$$

que se resuelve como:

$$\boldsymbol{h} = A^{-1}\boldsymbol{b} \tag{2.27}$$

Pero si $\boldsymbol{b} = 0$, es un sistema homogéneo de ecuaciones y no se puede resolver invirtiendo la matriz A. Entonces fijamos la escala de la homografía haciendo $h_{33} = 1$, y se obtiene de (2.24) lo siguiente:

$$\begin{aligned} h_{11}x_1 + h_{12}y_1 + h_{13} - h_{31}x_1x_2 - h_{32}y_1x_2 - x_2 &= 0 \\ h_{11}x_1 + h_{12}y_1 + h_{13} - h_{31}x_1x_2 - h_{32}y_1x_2 &= x_2 \end{aligned} \tag{2.28}$$

$$\begin{aligned} h_{21}x_1 + h_{22}y_1 + h_{23} - h_{31}x_1y_2 - h_{32}y_1y_2 - y_2 &= 0 \\ h_{21}x_1 + h_{22}y_1 + h_{23} - h_{31}x_1y_2 - h_{32}y_1y_2 &= y_2 \end{aligned} \tag{2.29}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_2 & -y_1x_2 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_2 & -y_1y_2 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \tag{2.30}$$

Ya que se tienen ocho incógnitas se requieren ocho ecuaciones y como se generan dos ecuaciones por cada correspondencia de puntos, se necesitan al menos cuatro correspondencias de puntos para poder calcular una homografía.

Los puntos deben de normalizarse a media cero y varianza uno, ya que de otra forma es inestable numéricamente.

$$\boldsymbol{p}'_1 = T_1\boldsymbol{p}_1, \quad \boldsymbol{p}'_2 = T_2\boldsymbol{p}_2 \tag{2.31}$$

$$\begin{aligned}
\lambda \boldsymbol{p}_2' &= H' \boldsymbol{p}_1' \\
\lambda T_2 \boldsymbol{p}_2 &= H' T_1 \boldsymbol{p}_1 \\
\lambda T_2^1 T_2 \boldsymbol{p}_2 &= T_2^{-1} H' T_1 \boldsymbol{p}_1 \\
\lambda \boldsymbol{p}_2 &= T_2^{-1} H' T_1 \boldsymbol{p}_1
\end{aligned} \tag{2.32}$$

y finalmente la homografía $H = T_2^{-1} H' T_1$
Falta poder resolver $A\boldsymbol{h} = \boldsymbol{b}$, para ello se puede utilizar la descomposición QR. Entonces, si $A = QR$:

$$\begin{aligned}
A\boldsymbol{h} &= \boldsymbol{b}, \\
QR\boldsymbol{h} &= \boldsymbol{b}, \\
Q^{-1}QR\boldsymbol{h} &= Q^{-1}\boldsymbol{b}, \\
Q^T QR\boldsymbol{h} &= Q^T \boldsymbol{b}, \\
R\boldsymbol{h} &= Q^T \boldsymbol{b}
\end{aligned} \tag{2.33}$$

Si se tiene $Q^T = \boldsymbol{c}$ y se supone una matriz r de 3×3:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \boldsymbol{c}_3 \end{bmatrix} \tag{2.34}$$

La última incógnita se cálcula como:

$$\begin{aligned}
r_{33}\boldsymbol{h}_3 &= \boldsymbol{c}_3 \\
\boldsymbol{h}_3 &= \frac{\boldsymbol{c}_3}{r_{33}}
\end{aligned} \tag{2.35}$$

para la segunda, se tiene que:

$$\begin{aligned}
r_{22}\boldsymbol{h}_2 + r_{23}\boldsymbol{h}_3 &= \boldsymbol{c}_2 \\
\boldsymbol{h}_2 &= \frac{(\boldsymbol{c}_2 - r_{23}\boldsymbol{h}_3)}{r_{22}}
\end{aligned} \tag{2.36}$$

y para la tercera:

$$\begin{aligned}
r_{11}\boldsymbol{h}_1 + r_{12}\boldsymbol{h}_2 + r_{13}\boldsymbol{h}_3 &= \boldsymbol{c}_1 \\
r_{11}\boldsymbol{h}_1 &= \boldsymbol{c}_1 - r_{12}\boldsymbol{h}_2 - r_{13}\boldsymbol{h}_3 \\
\boldsymbol{h}_1 &= \frac{(\boldsymbol{c}_1 - r_{12}\boldsymbol{h}_2 - r_{13}\boldsymbol{h}_3)}{r_{11}}
\end{aligned} \tag{2.37}$$

Una generalización de este proceso se muestra en el siguiente pseudocódigo:

```
h[N] = c[N]/R[N,N]
for (i = N-1; i>=1; i++){
         suma = 0.0;
         for(j=i+1; j<=N; j++)
                  suma+= R[i,j] * h[j]
         h[i] = (c[i] - suma) / R[i,i]
}
```

2.4. Calibración de la cámara una partir de homografías

Para recuperar información de una imagen es necesaria la calibración de la cámara, con esta se puede conocer la escala de un objeto en una escena y podemos hacer inferencias sobre las escalas de los demás objetos. Sin esta calibración podemos recuperar la forma de los objetos pero no su escala. [11].

De la suposición de que tenemos un plano y este está situado en el plano $x-y$, haciendo la sustitución en el modelo de la cámara oscura (2.10), donde $\lambda \mathbf{p} = \lambda \mathbf{p}_2 = M\mathbf{p}_1$ y siendo $\mathbf{p}$ los puntos, λ un factor de escala y M es una homografía. Podemos obtener los parámetros de la cámara oscura a partir de una homografía y esto es la autocalibración.

Para obtener los parámetros de la cámara oscura se relacionan las ecuaciones de la cámara oscura y de la homografía, de las ecuaciones (2.10) y (2.19), respectivamente, de lo cual se obtiene:

$$\lambda \boldsymbol{H} = K \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \tag{2.38}$$

Para calcular la homografía se tienen que normalizar los puntos a media cero y desviación estándar uno, esto porque en (2.38) K tiene unidades en milímetros o píxeles, $\boldsymbol{r}_{ij}$ en $[0, 1]$ y t en milímetros.

Se utiliza el método de Zhang [12] para el cálculo de la homografía. A partir de (2.38) se tiene que:

$$\begin{aligned} \lambda K^{-1} \boldsymbol{H} &= K^{-1} K \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \\ \lambda K^{-1} \boldsymbol{H} &= \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \end{aligned} \tag{2.39}$$

El método de Zhang usó restricciones sobre la matriz de rotación, esta debe ser una matriz ortogonal por lo que :

$$\begin{aligned} \boldsymbol{r}_1^T \boldsymbol{r}_2 &= 0 \\ |\boldsymbol{r}_1| = |\boldsymbol{r}_2| &= 1 \end{aligned} \tag{2.40}$$

Entonces (2.40) indica que $\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ son ortogonales (su producto punto es igual a cero), además ambos vectores tienen norma igual a 1.

A partir de (2.39)

$$\lambda K^{-1} \begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \tag{2.41}$$

$$\begin{aligned} \boldsymbol{r}_1 &= K^{-1}\boldsymbol{h}_1, \\ \boldsymbol{r}_2 &= K^{-1}\boldsymbol{h}_2, \\ \boldsymbol{t} &= K^{-1}\boldsymbol{h}_3. \end{aligned} \tag{2.42}$$

Estas tres ecuaciones se mantienen si λ ya se conoce. Si usamos la primera restricción (2.40):

$$\begin{aligned} \boldsymbol{r}_1^T \boldsymbol{r}_2 &= (K^{-1}\boldsymbol{h}_1)^T K^{-1}\boldsymbol{h}_2 = 0, \\ &= \boldsymbol{h}_1^T K^{-T} K^{-1} \boldsymbol{h}_2 = 0. \end{aligned} \tag{2.43}$$

aquí $(K^{-1})^T = K^{-T}$.

Si hacemos:

$$K^{-T}K^{-1} = W \tag{2.44}$$

Tenemos:

$$\boldsymbol{h}_1^T W \boldsymbol{h}_2 = 0 \tag{2.45}$$

De la matriz K :

$$\begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.46}$$

donde (u_0, v_0) se suponen en el centro de la imagen, por lo cual son conocidos, lo que hace falta conocer es la distancia focal f, de la cámara.

Para esto desarrollamos $W = K^{-T}K^{-1}$ y nos queda:

$$W = \frac{1}{f^2} \begin{bmatrix} 1 & 0 & -u_0 \\ 0 & 1 & -v_0 \\ -u_0 - v_0 & u_0^2 + v_0^2 + f^2 & w_{33} \end{bmatrix} \tag{2.47}$$

W es una matriz simétrica, donde el término que desconocemos es w_{33}.

De nuestra ecuación (2.45):

$$\boldsymbol{h}_1^T W \boldsymbol{h}_2 = \begin{bmatrix} \boldsymbol{h}_{11} & \boldsymbol{h}_{21} & \boldsymbol{h}_{31} \end{bmatrix} \begin{bmatrix} 1 & 0 & w_{13} \\ 0 & 1 & w_{23} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_{12} \\ \boldsymbol{h}_{22} \\ \boldsymbol{h}_{32} \end{bmatrix} = 0 \tag{2.48}$$

$$w_{33} = \frac{-\boldsymbol{h}_{12}(\boldsymbol{h}_{11} + \boldsymbol{h}_{31} w_{13}) - \boldsymbol{h}_{22}(\boldsymbol{h}_{21} + \boldsymbol{h}_{31} w_{23}) - \boldsymbol{h}_{32}\boldsymbol{h}_{11} w_{13} - \boldsymbol{h}_{32}\boldsymbol{h}_{21} w_{23}}{\boldsymbol{h}_{32}\boldsymbol{h}s_{31}} \tag{2.49}$$

Ya que $w_{33} = u_0^2 + v_0^2 + f^2$ entonces podemos calcular $f^2 = w_{33} - u_0^2 - v_0^2 = a$ y si $a > 0$: $f = \sqrt{a}$.

Conociendo f podemos conocer la matriz K,

$$\lambda K^{-1}[\boldsymbol{h}_1\ \boldsymbol{h}_2\ \boldsymbol{h}_3] = A = [\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{t}]$$

El factor de escala se puede calcular como:

$$\begin{aligned} \lambda_1 &= |\boldsymbol{a}_1|, \\ \lambda_2 &= |\boldsymbol{a}_2|, \\ \lambda &= \frac{(\lambda_1 + \lambda_2)}{2} \end{aligned}$$

y la matriz de rotación se calcula como:

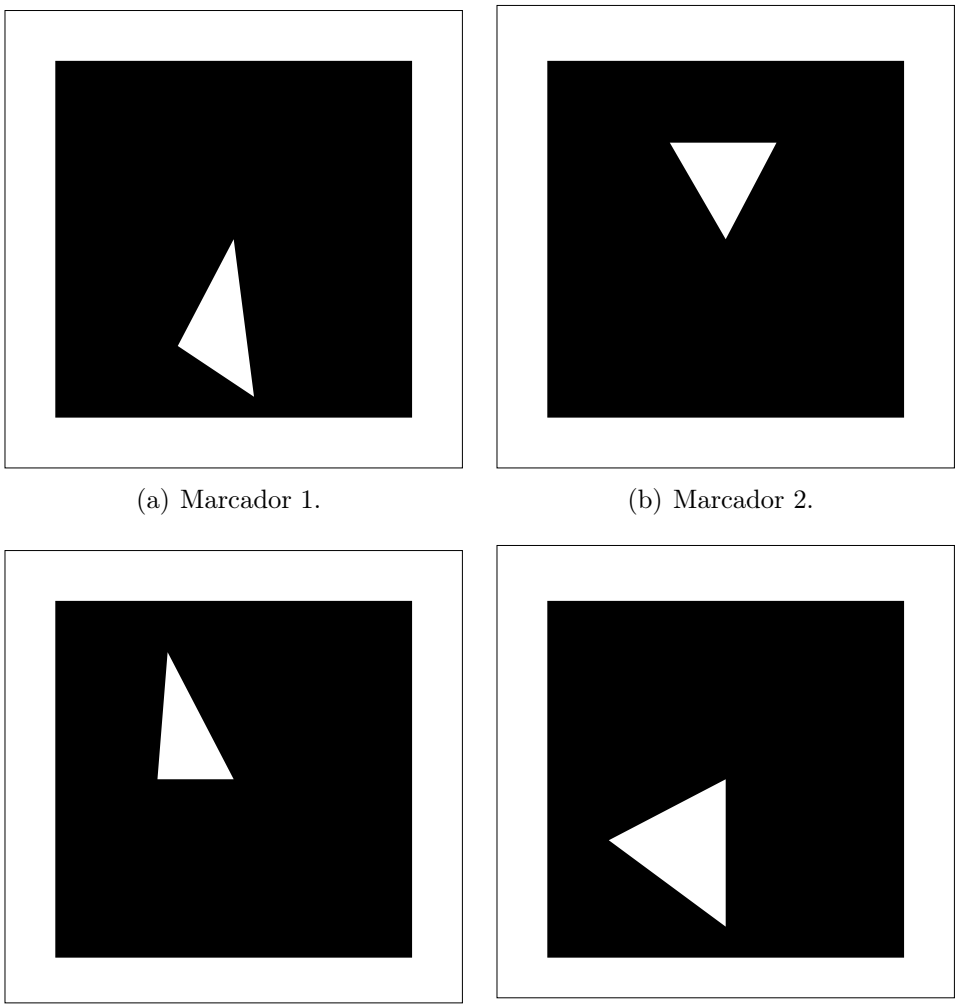
$$\begin{aligned} \frac{1}{\lambda} A &= [\boldsymbol{r}_1\ \boldsymbol{r}_2\ \boldsymbol{t}] \\ \boldsymbol{r}_3 &= \boldsymbol{r}_1 \times \boldsymbol{r}_2 \end{aligned}$$

Conociendo $\boldsymbol{r}_1, \boldsymbol{r}_2$ y $\boldsymbol{r}_3$ ya tenemos a R' que no es ortogonal y tenemos que ortogonalizarla, para lo cual se usa la descomposición en valores singulares (DVS):
$R' = UDV^T$, U y V son matrices ortogonales y D es una matriz diagonal. $R = UV^T$.
R ya es ortogonal.

2.5. Marcadores basados en tipo de orden

Un marcador de referencia o fiduciario es un elemento único que facilita su identificación y se conoce su forma antes de la captura, cuando es capturado se reconoce debido a un entrenamiento previo o algún algoritmo de identificación. Es utilizado en los sistemas de RA para escalar la imagen o relacionar imágenes independientes.

El tipo de orden (OT) describe un conjunto de puntos que evita el uso de información métrica. OT es un descriptor invariante a las transformaciones geométricas



(a) Marcador 1. (b) Marcador 2.

(c) Marcador 3. (d) Marcador 4.

Figura 2.6: Marcadores utilizados.

euclidianas, cambio de escala y proyección de perspectiva. Por lo que es usado para construir marcadores fiduciales [13].

Los marcadores utilizados en esta tesis fueron marcadores de seis puntos. Esta cantidad de puntos se escogió porque resulta en una perturbación máxima [14] de 10.61, este valor significa que dos puntos tienen que moverse 10.61 unidades (con respecto al valor de la secuencia de los puntos) para cambiar su tipo de orden. En la figura 2.6 se muestran los marcadores utilizados.

La matriz lambda es una forma de representar el tipo de orden para un conjunto de puntos. El algoritmo para calcular la matriz lambda incluye los siguientes pasos:

- Se tiene la matriz de tamaño $n \times n$ para n puntos,

- n columnas de los puntos $\mathbf{p}_1$, $\mathbf{p}_2$, ... $\mathbf{p}_n$,
- n renglones de los puntos $\mathbf{p}_1$, $\mathbf{p}_2$, ... $\mathbf{p}_n$,
- se traza una línea entre los puntos $\mathbf{p}_i$ y $\mathbf{p}_j$ $(i \neq j)$,
- y se cuentan los puntos que están a la izquierda de la línea.

Esto se hace para cada punto de la cubierta convexa del conjunto de puntos, por lo que para los marcadores de seis puntos se obtienen cuatro matrices lambda. El cálculo de la matriz lambda para los puntos de la figura 2.7, que corresponden a la figura 2.6(a), se muestra a continuación.

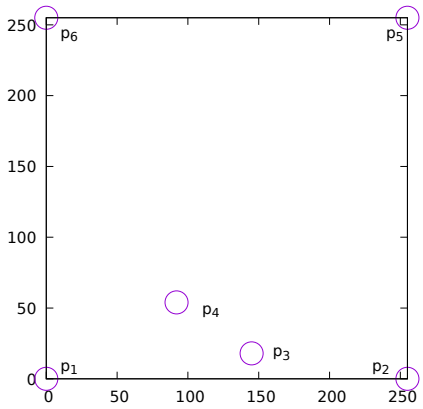


Figura 2.7: Puntos del marcador.

La primera matriz lambda queda de la siguiente manera. La primera columna y el primer renglón muestran los índices a la matriz lambda.

	1	2	3	4	5	6
1	-	4	3	2	1	0
2	0	-	1	2	4	3
3	1	3	-	1	3	2
4	2	2	3	-	2	1
5	3	0	1	2	-	4
6	4	1	2	3	0	-

La figura 2.8, muestra el etiquetado para la obtención de las tres matrices lambda restantes, correspondientes al marcador.

Debido a la relación de la parte superior de la matriz con la parte inferior, $\lambda_{i,j} + \lambda_{j,i} = n-2$, para $i \neq j$, no es necesario calcular las matrices completas. A continuación se muestra el cálculo de las matrices restantes.

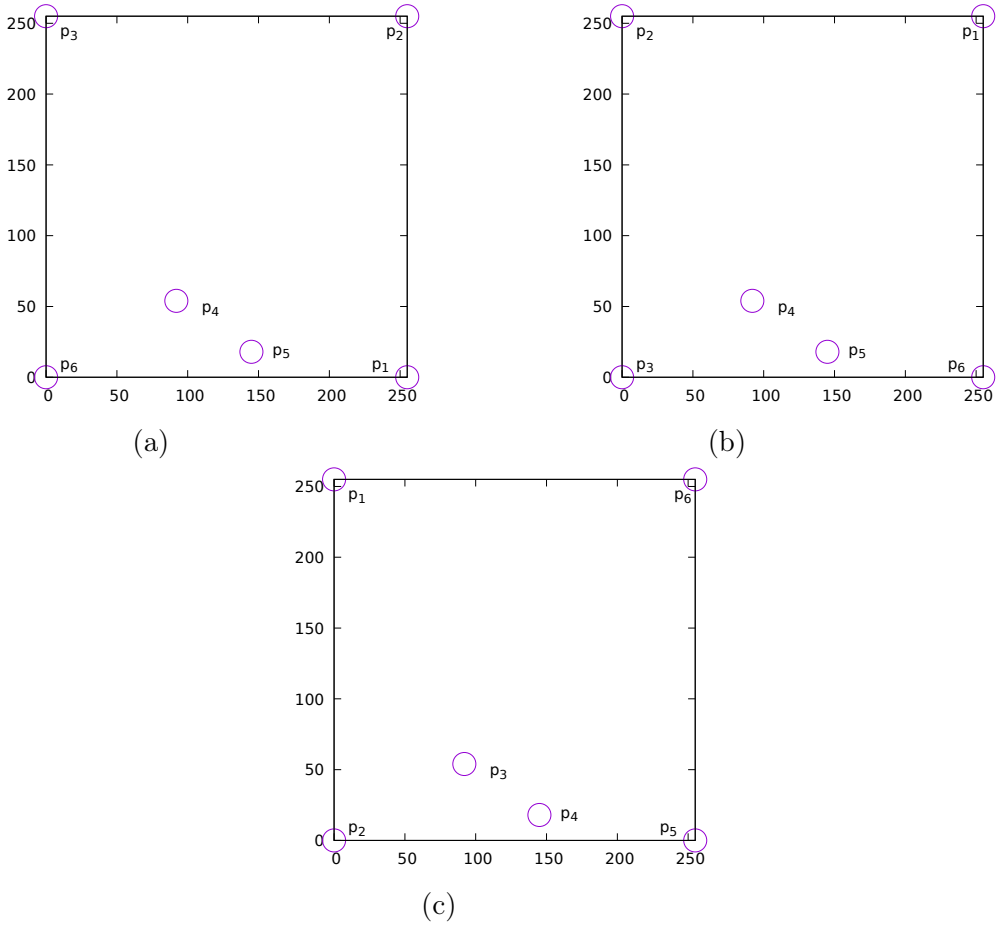


Figura 2.8: Etiquetado para las tres matrices lambda restantes.

Con respecto a p_1 respecto al etiquetado de la figura 2.8(a).

	1	2	3	4	5	6
1	-	4	3	2	1	0
2		-	4	2	1	3
3			-	3	2	4
4				-	3	2
5					-	1
6						-

Con respecto a p_1 respecto al etiquetado de la figura 2.8(b).

	1	2	3	4	5	6
1	-	4	3	2	1	0
2		-	4	3	2	1
3			-	2	3	4
4				-	3	2
5					-	3
6						-

Con respecto a p_1 respecto al etiquetado de la figura 2.8(c).

	1	2	3	4	5	6
1	-	4	3	2	1	0
2		-	2	3	4	1
3			-	3	2	2
4				-	3	3
5					-	4
6						-

Ahora se debe obtener la mínima lexicográfica, escribimos el resultado de las matrices en fila para compararlas e identificar la mínima.

4	3	2	1	0	1	2	4	3	1	3	2	2	1	4
4	3	2	1	0	4	2	1	3	3	2	4	3	2	1
4	3	2	1	0	4	3	2	1	2	3	4	3	2	3
4	3	2	1	0	2	3	4	1	3	2	2	3	3	4

De aquí la primera es la mínima lexicográfica, es la que se almacenaría para poder identificar el marcador posteriormente.

Las coordenadas de los puntos de los marcadores 2.6(a), 2.6(b), 2.6(c) y 2.6(d) son,

Marcador 1

x	**y**
0	0
0	255
255	255
255	0
145	18
92	54

Marcador 2

x	**y**
0	0
0	255
255	255
255	0
90	199
165	199

Marcador 3

x	**y**
0	0
0	255
255	255
255	0
80	221
81	127

Marcador 4

x	**y**
0	0
0	255
255	255
255	0
44	86
127	25

Las matrices lambda mínimas para los tres últimos marcadores son:

Marcador 2:	4	3	2	1	0	2	1	4	3	2	3	2	2	1	4
Marcador 3:	4	3	2	1	0	1	3	4	2	2	3	1	2	3	4
Marcador 4:	4	3	2	1	0	1	3	4	2	3	2	1	3	3	4

2.6. Detección del marcador

Para realizar la detección del marcador, ya que se usarón marcadores en tipo de orden, lo primero que se hace es un procesamiento de imagénes, el cual incluye:

1. Se transforma la imagen a escala de grises,
2. se aplica un filtro gausiano de tamaño 7×7, que es un filtro de suavizado,
3. se aplica un umbralizado global para producir una imagen binaria,
4. y se obtienen las componentes conectadas de la imagen binarizada.

Para cada componente conectada se realiza:

1. Si la componente toca el borde de la imagen, se desecha;
2. si la componente tiene cuatro esquinas, entonces este es el marcador.

Posteriormente se realiza el reconocimiento en la componente conectada del marcador, para lo cual se reconoce el triángulo dentro del marcador. Los valores de los cuatro vértices del cuadrado, y dos vértices del triángulo (se desecha el vértice más cerca del centro del marcador) son los seis puntos que forman el marcador y con ellos se calcula su matriz lambda, tal como se explicó en la sección anterior.

Se deben tener almacenadas las matrices lambda pertenecientes a cada secuencia de puntos (cada marcador) y cuando se reconoce un marcador se verifica de cual de ellos se trata.

2.7. Detección de colisiones

La detección de colisiones es un método que fue desarrollado en el campo de los videojuegos [15] para detectar si dos objetos han colisionado. Esta detección se puede realizar por área o pixel a pixel.

El método que se utilizó en esta tesis fue la detección por área, en la cual los objetos ocupan un área, rectangular o esférica y cuando estas áreas se superponen existe una colisión.

En esta tesis se utilizó un área esférica, de la cual tenemos que los centros de las esferas y sus radios están dados por:

$$c_1 = [x_1, y_1, z_1]^T, r_1$$
$$c_2 = [x_2, y_2, z_2]^T, r_2$$

Existe un colisión entre esas dos esferas si:

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 < (r_1 + r_2)^2$$

Capítulo 3

Sistema de Realidad Aumentada

En el presente capítulo se exhibe el desarrollo del sistema de realidad aumentada. Se muestra la generación del modelo del pez y las imagénes utilizadas como referencia para su desarrollo. Se explica el proceso de animación y la ecuación utilizada para ésta, además se muestran los comportamientos y estados implicados en la animación, así como los diagramas de flujo de este proceso. Por otro lado, se presenta el espacio de trabajo que fue requerido para llevar a cabo esta tesis. Se explica también el uso y configuración del sensor.

3.1. Generación del modelo de un pez

Para la generación del modelo del pez, se tenían dos opciones. La primera, buscar un modelo de pez ya hecho, que fuera de licencia libre, y así, solo integrarlo al sistema. El inconveniente de esta opción, es que no se tiene el control absoluto del modelo y por lo tanto, sería díficil adaptarlo a la animación. La segunda opción fue hacer un modelo que cumpliera con las características físicas requeridas para la integración al sistema y su animación. Se optó por hacer un modelo propio, para lo cual se eligió el software libre de modelado 3D Blender [16, 17] cuyo logo se muestra en la figura 3.1. Este software es gratuito y está directamente disponible en todas las distribuciones de GNU/Linux, para MacOS y Windows.

Figura 3.1: Logo del software de modelado Blender.

En Blender se tiene el sistema de coordenadas cartesianas de mano izquierda con el eje z apuntando hacia arriba [16, blenderWiki]. De tal manera que el eje x se visualiza de izquierda a derecha, el eje y de adelante hacia atrás, y el eje z de abajo hacia arriba. El sistema de coordenadas global se puede ver en la figura 3.2.

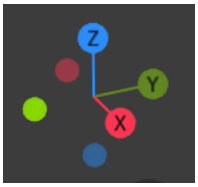
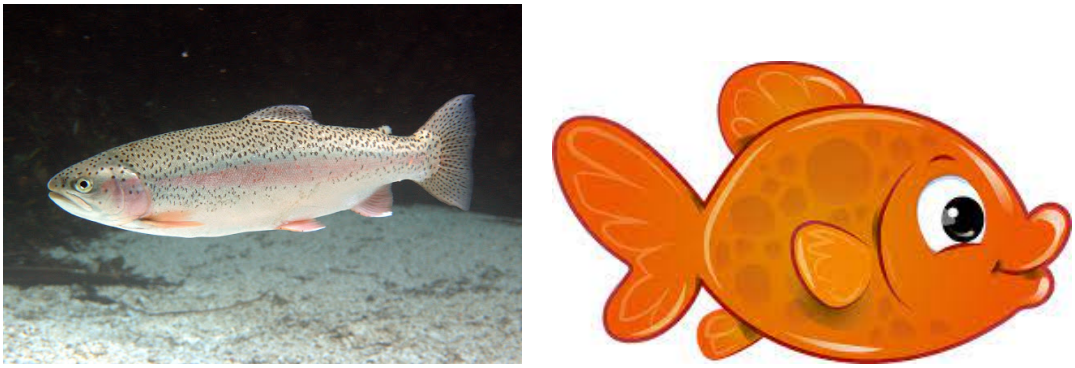


Figura 3.2: Sistema de coordenadas cartesianas de Blender.

El modelado de los objetos se realizó usando como referencia imágenes planas de dos peces, un pez naranja tipo caricatura y una trucha, los cuales se pueden ver en la figura 3.3.



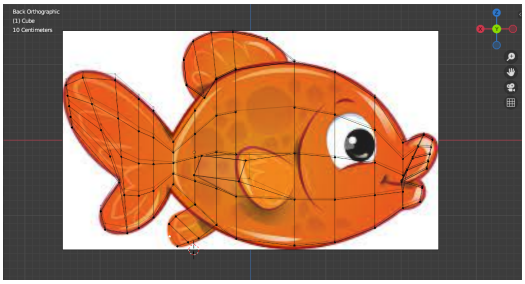
(a) Imagen de una trucha. (b) Imagen de un pez animado.

Figura 3.3: Imagénes de referencia para el modelado de los objetos virtuales.

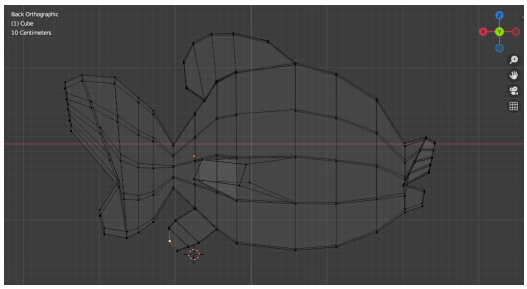
La figura principal utilizada, fue un cubo [18], a partir de este se realizó todo el modelo del pez, haciendo extensiones de los vértices para crear nuevas caras con nuevas aristas y escalando estas. En la figura 3.8 podemos observar estos pasos del modelado de uno de los objetos.

El modelo de la trucha se observa en la figura 3.5, en esta vista se ve cada uno de los vértices, aristas y caras del objeto.

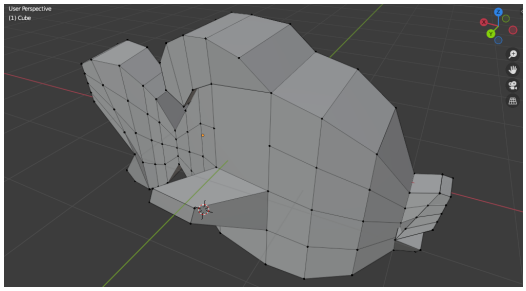
Una vez que se tuvo el modelo hasta este punto, se realizó el mapeo de coordenadas de textura [19], en el cual se utiliza una imagen y se indica una correspondencia entre cada una de las caras y una coordenada (u, v) de la imagen de textura a utilizar.
En la figura 3.6 del lado derecho se muestra la imagen que se tomo como textura para el modelo del pez, del lado izquierdo se aprecio el modelo, ya con esta textura implementada.



(a) Imagen de referencia, proyectada en los ejes $x - z$, vista frontal.



(b) Vista de alambre del objeto, sin la imagen de referencia.



(c) Vista sólida del objeto 3D, sin la imagen de referencia.

Figura 3.4: Pasos principales del modelado del pez.

3.2. Animación de un pez

La técnica de animación utilizada es la llamada animación procedural o animación generada por computadora [20], la cual es una técnica en la que se utiliza un método algoritmico para calcular las variables que definen el comportamiento del objeto a lo largo del tiempo [21].

Para la animación del pez se realizó un estudio acerca del movimiento de los objetos articulados, específicamente de peces. Se encontraron dos formas de simular el movimiento de este:

1. Mediante el uso de fórmulas basadas en la física del movimiento real de los peces.

2. Utilizando un conjunto de fórmulas sencillas que logran describir el comportamiento del pez.

Utilizar las fórmulas basadas en la física del movimiento presenta un mayor costo computacional que el que se obtiene de utilizar fórmulas sencillas para describir el moviento del pez. Es por ello que se utilizó está segunda opción, ya que los requerimientos del sistema no restringen al uso de la física del movimiento real, pues el

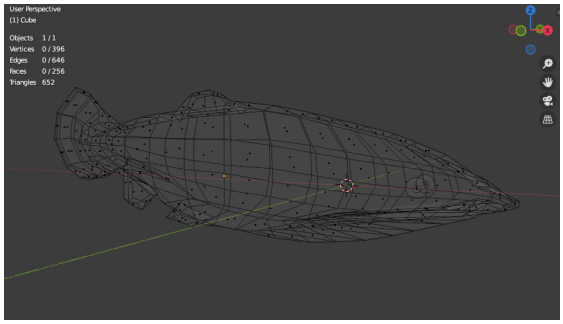


Figura 3.5: Modelo de la trucha. Vértices, aristas y caras.

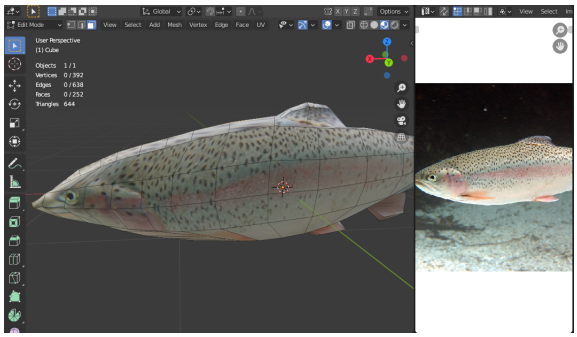


Figura 3.6: Texturizado del modelo del pez.

objetivo es poder ver un movimiento que parezca real aunque este no venga de la física que lo describe.

3.2.1. La ecuación de forma de onda

Dentro de la investigación relizada se encontró un trabajo que utiliza animación esquelética para movimiento de peces artificiales, a través de la técnica de animación procedimental, modelando el movimiento por medio de una formulación numérica simple [22]. El modelo matemático de dicha investigación incluye, entre otras cosas, el uso de la función seno, esto sirvió como idea para generar un modelo propio, utilizando dicha función.

La onda senoidal puede observarse como una oscilación repetitiva, este comportamiento es de gran similitud con el movimiento del nado de un pez, visto este desde el plano $x-y$ donde el plano z es el plano hacia arriba, como se muestra en la figura 3.7.

La forma general de una onda senoidal está representada en la ecuación (3.1):

$$f(t) = A\,\text{sen}\left(\frac{2\pi vt}{\lambda} + \phi\right) \tag{3.1}$$

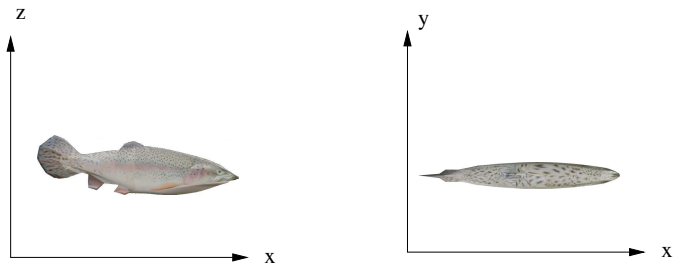


Figura 3.7: Referencia de la vista lateral(x-z) y superior(x-y).

donde: ϕ es el desplazamiento de fase. v es la velocidad. t el tiempo y λ es la longitud de onda. A es la amplitud de la senoidal. Además,

$$v = \frac{\lambda}{t}. \tag{3.2}$$

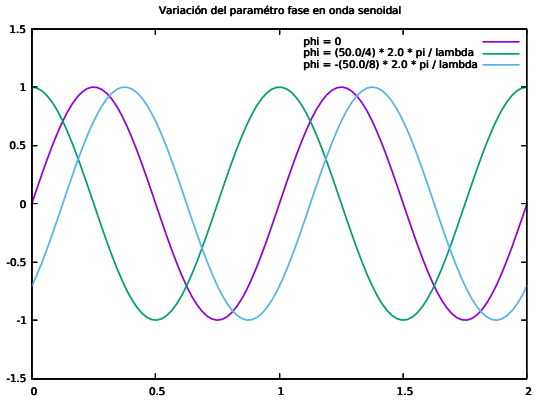
La variación de los paramétros de fase, amplitud y velocidad, de la onda senoidal, permitió simular el movimiento requerido para el nado del pez. A continuación se muestran algunas pruebas de la variación de dichos paramétros. Se relizarón simulaciones para los dos modelos de pez que se diseñaron. Estas simulaciones consisten realizar los cálculos y gráficar las posiciones en las que se encontrará cada parte del pez a lo largo de tiempo. Esto sirvió para visualizar el movimiento y de esa manera ajustar los valores del modelo de forma que se asemejará al movimiento real del nado de una trucha. La simulación del movimiento de la trucha, se observa en la figura 3.9. Para este modelo de movimiento se obtuvieron los siguientes valores. $l = 50$. $v = 180$. $\lambda = 1.8 * l$. $A = 5$

La simulación del movimiento del pez naranja o pez animado se muestra en la figura 3.10. Para este modelo de movimiento el único valor que cambió fue el de amplitud, esta vez con A = 1.5. Es importante mencionar que en este caso el modelo solo fue aplicado en la parte trasera del pez, específicamente en su cola, pues el tipo de dibujo animado con el que se trabajó así lo requirió.

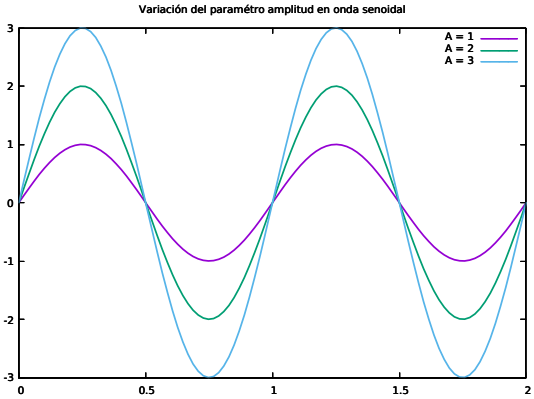
3.3. Estados

Se definió un estado como un movimiento básico de la animación que posteriormente puede formar parte de una secuencia de estados que describan un comportamiento más complejo. Se tienen tres estados principales:

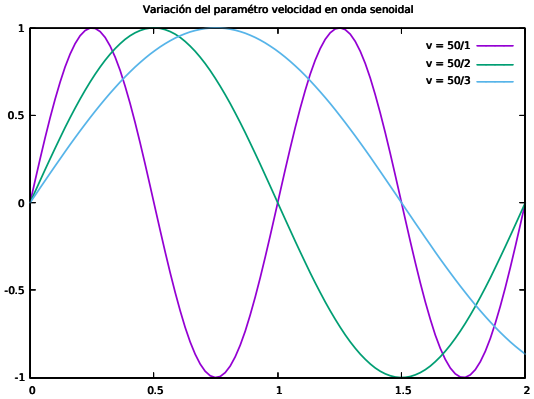
- E1: nado sin desplazamiento; Este es ejecutado a través del modelo presentado en la sección anterior, aplicando la fórmula a lo largo del objeto virtual, de forma horizontal.



(a) $\phi = 0, 2\pi(50/4)/50, 2\pi(-50/8)/50$



(b) $A = 1, 2, 3$

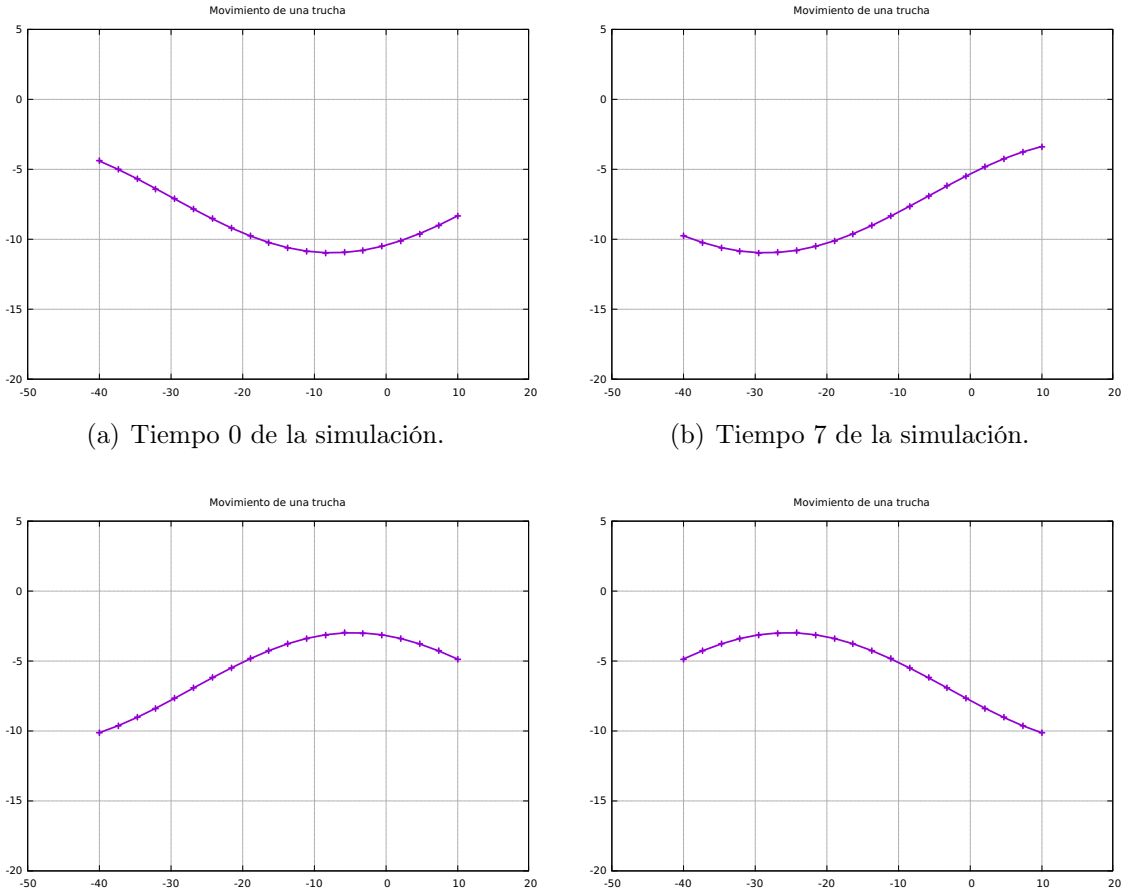


(c) $v = 50/1, 50/2, 50/3$

Figura 3.8: Pruebas de variación de fase, amplitud y velocidad de la onda senoidal.

- E2: nadar con desplazamiento; Este estado se logra realizando una traslación del objeto por medio de openGL, al mismo tiempo que se ejecuta el nado sin desplazamiento.

- E3: dar la vuelta; Este estado es un caso especial en el que se combinan dos modelos, una para el giro en si mismo y el de nado con desplazamiento para los momentos de transición.
  El giro requiere una mayor amplitud en la onda senoidal para que se observe que se arquea el cuerpo del pez al dar la vuelta. En este caso es necesario mantener una sincronia entre el modelo del giro y el modelo del nado con desplazamiento, pues se debe hacer el cambio durante los tiempos en que los modelos se cruzan, es decir cuando estos modelos pasan por cero se puede cambiar uno por otro, esto con el objetivo de que la animación no se vea cortada por algún salto entre los modelos y para que una vez concretado el giro se regrese al estado de nado



(a) Tiempo 0 de la simulación.

(b) Tiempo 7 de la simulación.

(c) Tiempo 14 de la simulación.

(d) Tiempo 21 de la simulación.

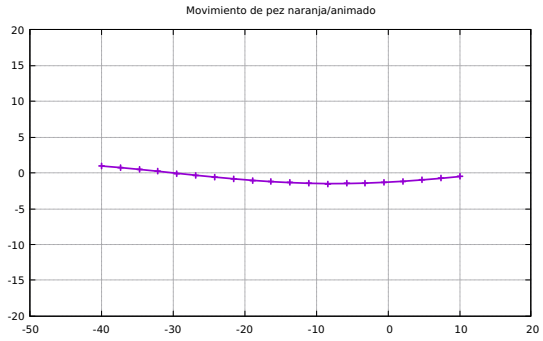
Figura 3.9: Simulación del movimiento de una trucha, los parámetros para esta son: $l = 50$. $v = 180$. $\lambda = 1.8 * l$. $A = 5$.

con desplazamiento.

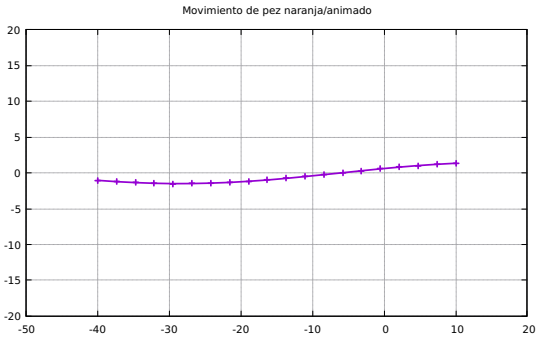
3.4. Comportamientos

Los comportamientos están compuestos por una serie de estados que juntos describen un movimiento complejo. Los comportamientos definidos son:

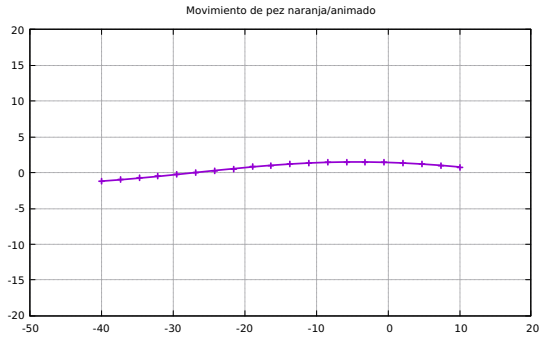
1. Estático : Definido solo a tráves del E1, nado sin desplazamiento.
2. Huir : Se ejecuta realizando un número de giros al azar. El E2, nado con desplazamiento y otra serie de giros (E3) para que vuelva a mirar a la vista principal.
3. Regresar: Este comportamiento realiza un nado con desplazamiento E2.



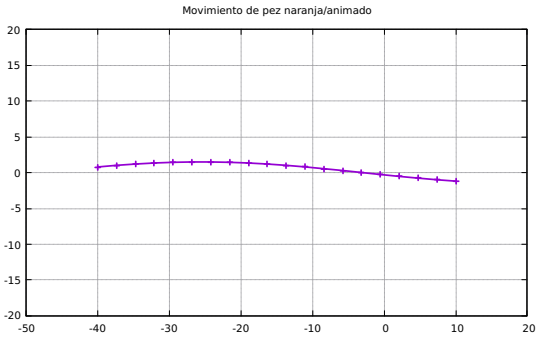
(a) Tiempo 0 de la simulación.



(b) Tiempo 7 de la simulación.



(c) Tiempo 14 de la simulación.



(d) Tiempo 21 de la simulación.

Figura 3.10: Simulación del movimiento del pez naranja/animado.

Todos estos comportamientos son ejecutados ante distintos tipos de eventos de interacción detectados por medio del sensor.

3.5. Zona de trabajo

Se adecuó una zona de trabajo especial para el uso y visualización del sistema, debido a los requerimientos necesarios de éste se necesitó un espacio en el que pudiera estar el equipo donde se ejecuta el software con la pantalla de visualización y que a su vez se tuviera espacio para hacer uso del sensor magnético para realizar la interacción. Este espacio de trabajo se instaló en una mesa de plástico. El material de la mesa es parte fundamental para el correcto funcionamiento del sensor ya que el este funciona utilizando una fuente magnética, con lo cual los datos se ven severamente afectados cuando el sensor es utilizado cerca de objetos metálicos.

Ya que se tenía la mesa, se diseñó un dispositivo de cartón para fijar la distancia entre la fuente magnética del sensor y el marcador que se utilizó. Se muestran las medidas iniciales de este dispositivo en la figura 3.11, donde el cuadro pequeño, de 5.5×5.9 cm. es un orificio en el cual se fija la fuente magnética del sensor y el cuadro más grande de 9×9 cm. es donde se coloca el marcador.

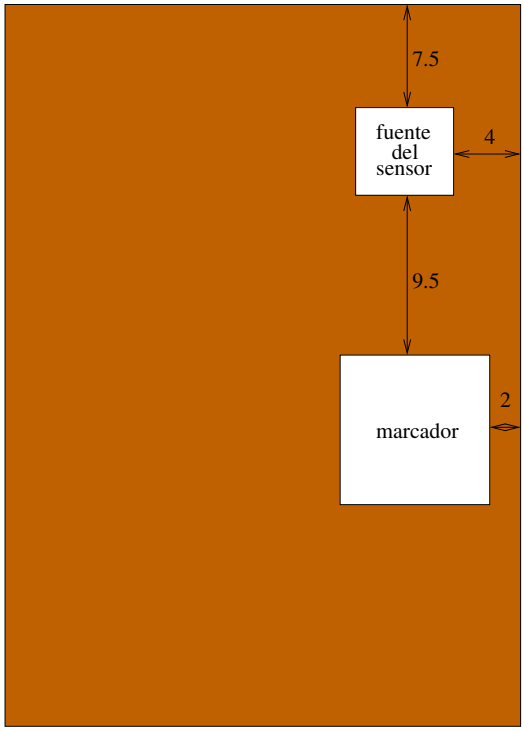


Figura 3.11: Dispositivo para fijar la distancia entre el sensor y el marcador, las distancias están dadas en cm.

3.6. Sensor para la interacción

El sensor utilizado es el Patriot de la marca Polhemus, el cual es un dispositivo de seguimiento de seis grados de libertad, este funciona a través de una fuente magnética la cual emite un campo electromagnético y los sensores dentro del rango de alcance del campo se rastrean, entregando así los datos de posición y orientación del sensor [23]. En la imagen 3.12 [4] se muestran los elementos del dispositivo.

La fuente magnética tiene definidos sus ejes x, y y z, los cuales son el origen de referencia inicial para las mediciones, por ello, esta fuente debe ser fijada en una posición inamovible. Por otro lado, el sensor mide el campo magnético generado por la fuente y es utilizado para obtener la posición y orientación de éste. El sensor utilizado es uno tipo pluma llamado *Stylus*. La unidad electrónica del sistema tiene los conectores y controladores de entrada y salida necesarios para para interconectar

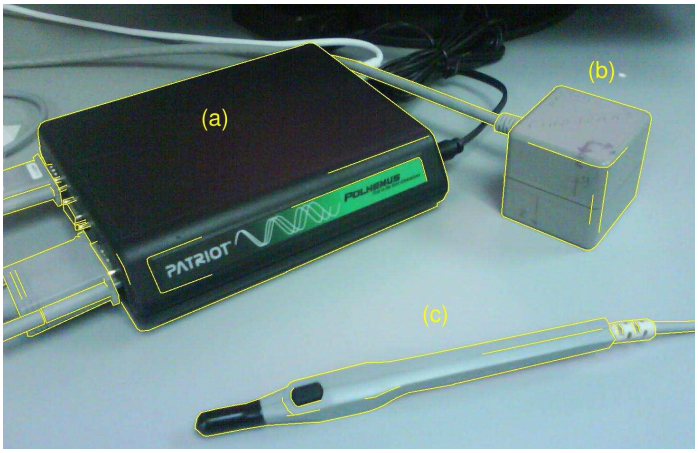


Figura 3.12: Dispositivo Patriot: (a) la unidad electrónica del sistema, (b) la fuente magnética y (c) el sensor.

el sensor, la fuente magnética y el puerto USB y RS-232 con los que se comunica a la computadora. Además, la unidad electrónica, es la que se encarga de los cálculos pertinentes para la entrega de datos, en este sentido es importante mencionar que la comunicación con el dispositivo se realiza por medio de comandos en formato ASCII y las respuestas en formato binario o ASCII, la entrega de datos puede ser realizada tanto en centimetros como en pulgadas según sea configurado.

3.6.1. Instalación y configuración del sensor

Como se mencionó en la sección anterior, el dispositivo Patriot al funcionar por medio de un campo electromagnético se ven afectados los resultados cuando se coloca cerca de objetos metálicos, por ello se utilizó una mesa completamente de plástico para colocarlo. Posteriormente se realizó la configuración del dispositivo, este se conectó por medio del puerto RS-232 utilizando un cable adaptador USB hacia la computadora, ya que ésta no cuenta con puerto RS-232.

La configuración básica de comunicación del dispositivo, por medio del puerto RS-232, se realizó utilizando el software minicom, con los parámetros especificados en el manual del dispositivo [24], en la imagen 3.13 se muestra esta configuración.

Una vez realizada esta configuración se obtuvieron los primeros datos al enviar la petición de información con el comando "p", el cual como respuesta entrega una fila con siete columnas de datos donde se puede ver:

1. Número del dispositivo
2. Coordenada x
3. Coordenada y
4. Coordenada z

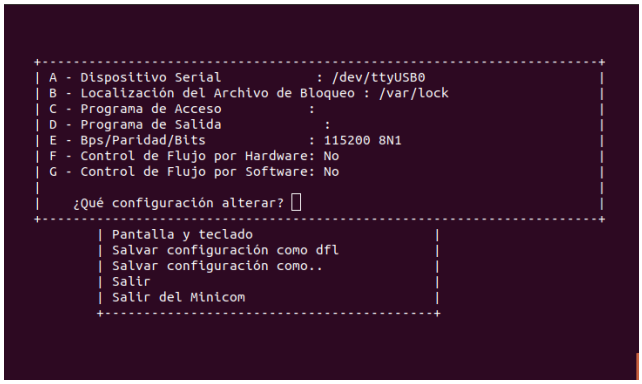


Figura 3.13: Configuración de la conexión del dispositivo a través del puerto serial RS-232 con el software minicom.

5. Ángulo de orientación *Acimuth*

6. Ángulo de orientación *Elevación*

7. Ángulo de orientación *Rotación propia*

Un ejemplo de estos datos se observa en la imagen 3.14. Posteriormente se realizó

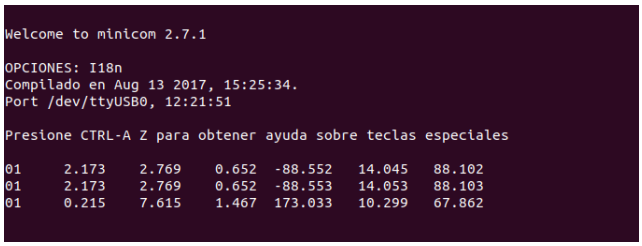


Figura 3.14: Obtención inicial de datos del sensor.

esta misma configuración a través de un programa en lenguaje C, para agregarlo más adelante al sistema. Dicha configuración requirió de la biblioteca *termios.h* [25] la cual contiene atributos para el control de puertos de comunicación. Lo primero que se realizó fue definir una variable de comunicación con el puerto, después se realizaron los cambios en la configuración de la estructura termios [26][27], estos cambios fueron los mismos realizados en el software de minicom y se describen a continuación: Bits por segundo : 115200. Bits de datos : 8. Paridad : No. Bits de parada : 1. Control de flujo : Ninguno.
Una vez realizada la asignación de estos valores en la estructura *termios*, se puede comenzar la comunicación con el sensor. En este caso se realizó la configuración de las unidades de respuesta para que estas fueran en centímetros. Finalmente se puede realizar la petición y lectura de los datos sensados. A continuación se muestra parte del código de este procedimiento.

```
int main(int argc, char const *argv[]) {
  int serialPort = open( "/dev/ttyUSB0", O_RDWR );
  if ( (serialPort < 0)) {
    fprintf( stderr, "Error connecting to tracker.\n");
    fprintf( stderr, "Error %i from open: %s\n", errno,
       strerror(errno));
    exit ( -1 );
  }

  struct termios newtio;
  newtio.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
  newtio.c_iflag = IGNPAR;
  newtio.c_oflag = 0;       // Raw output
  newtio.c_lflag = 0;       // Set input mode
  newtio.c_cc[VTIME] = 10;  // Inter-character timer
  newtio.c_cc[VMIN] = 0;    // No blocking read

  int br = 0;
  char buf[2000];
  // Configuracion del sistema en centimetros
  write( serialPort, "u1\r", 3 );
  while(1){
        //Peticion de datos
    write( serialPort, "p", 1 );
    sleep(1);
    //Lectura de la respuesta
    br = read( serialPort, buf, 2000 );
    if (br > 0) {
      printf("Bytes leidos: %d\n", br);
      printf("%s\n",buf);
    }

  }
  //Cierre del puerto serial
  int v_close = close(serialPort);
  if (v_close != 0) {
    printf("No cerro bien serialPort!!\n ");
  }
  return 0;
}
```

Ya con los datos obtenidos del sensor, se realizaron pruebas de precisión, para determinar la correspondencia entre los datos del sensor y la distancia que se media de forma física entre la fuente del dispositivo y el sensor *Stylus*. Se utilizó una regla en centímetros para tomar los datos con intervalos de un centimetro desde la fuente

magnética y alejando la medición a lo largo de la regla. A continuación la imagen 3.15 muestra como fueron realizadas estas mediciones. Inicialmente se contaba con

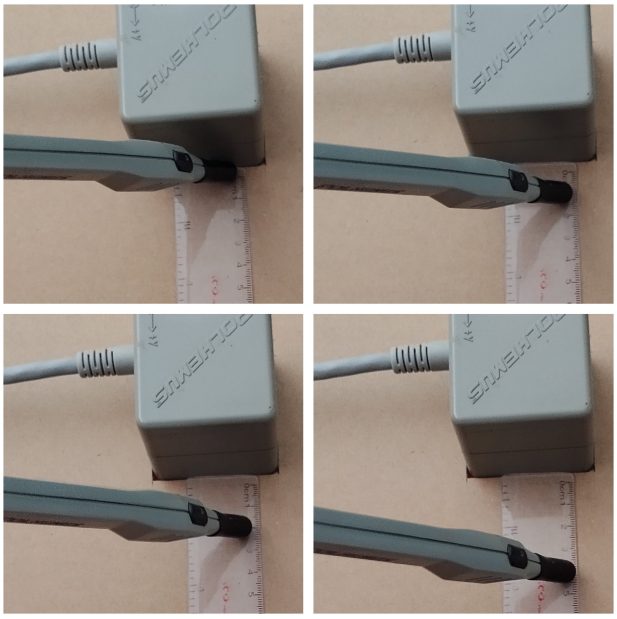


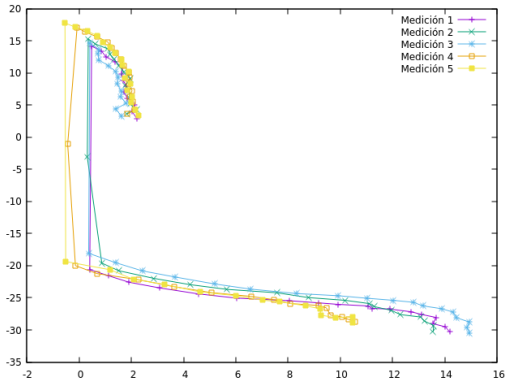
Figura 3.15: Obtención de medidas de prueba para conocer la presición del sensor.

una mesa de madera y patas de metal, con ésta se realizaron las primeras pruebas. Fueron 30 mediciones, 5 veces, de los cuales se obtuvieron los datos de la figura 3.16, los cuales se puede observar son bastante irregulares.

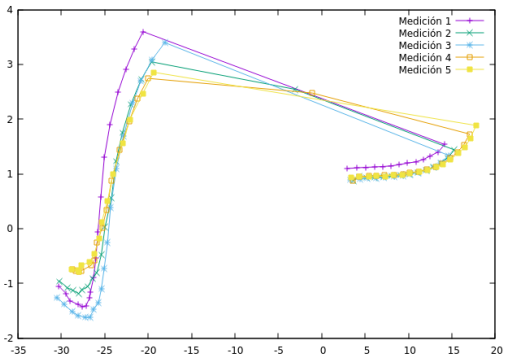
Los datos de la gráficas anteriores mostraron un deterioro de presición en ciertas partes de las distancias que se tomaban como referencia, por ello se consiguió la mesa de plástico con la cuál, se realizaron nuevamente la mismas pruebas, y esta vez, los datos obtenidos fueron mejores, pues conservaban su linealidad respecto a la distancia esperada. En la figura 3.17 se muestran estos datos.

3.7. Comportamiento general del pez

El comportamiento del pez es una serie de pasos que dependen de la interacción del sensor con el objeto virtual (pez). Estos pasos se ejecutan como combinaciones de los estados y comportamientos definidos en las secciones 3.3 y 3.4 respectivamente. A continuación se muestra el diagrama de flujo principal, figura 3.18, en el cual se observa que el pez tiene un comportamiento inicial donde se encuentra en un estado de nado sin desplazamiento, pero una vez que se presenta la interección a través del sensor, detectada con la colisión del sensor y el objeto (el pez), este último tiene un comportamiento de huida al sensor.



(a) Gráfica de las coordenadas $x - y$.

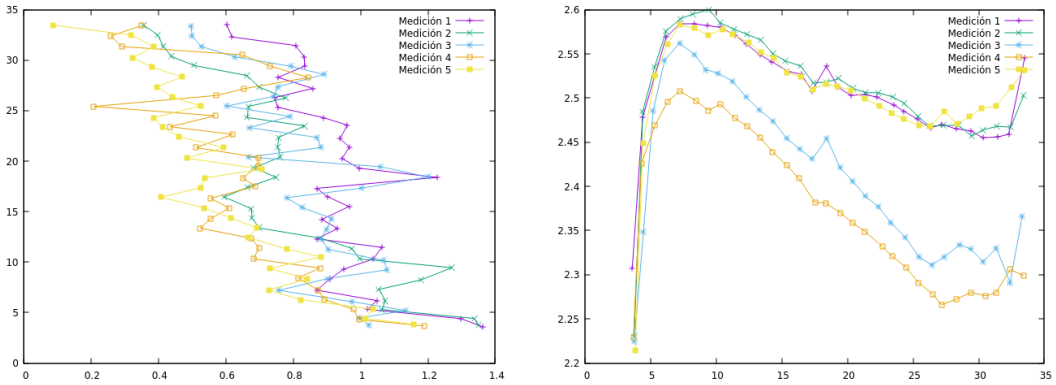


(b) Gráfica de las coordenadas $y - z$.

Figura 3.16: Pruebas de las mediciones: 5 conjuntos de 30 mediciones cada uno, ambos ejes representan centímetros, estas mediciones tienen interferencia por una mesa de metal.

La máquina de estados mostrada en la figura 3.19 muestra de forma detallada lo que se realiza durante el proceso de "Huir". Se puede observar que este proceso está formado de un inicio, un fin y cuatro etapas intermedias. La primera etapa es una serie de entre 5 y 8 giros aleatorios (izquierda o derecha), que el terminar permite ir a la siguiente etapa, donde se realiza un nado con desplazamiento. En la etapa tres se validan y generan las condiciones y los datos necesarios, nuevos números aleatorios, que servirán para la última etapa, donde se realiza nuevamente una serie de entre 5 y 8 giros aleatorios con el objetivo de volver a posicionar el pez mirando hacia la cámara, ya que durante los primeros giros realizados este queda "dando la espalda" a la cámara. El diagrama de flujo de la figura 3.20 es una implementacion de la máquina de estados del comportamiento huir con sentancias if.

La imagen 3.22 muestra un diagrama de estados del proceso de giro aleatorio describiendo las acciones realizadas durante esta etapa y se presenta un diagrama de flujo del mismo en la figura 3.21, este proceso aumenta el contador de los giros realizados al momento de su ejecución, valida si el contador de las vueltas es mayor al número de vueltas requerido y detiene el proceso de giro, modificando el estado huir en el que se encuentra el proceso general. Si la condición no se cumple continua haciendo girar el pez, a la derecha o izquierda, según sea el caso.



(a) Gráfica de las coordenadas $x - y$.

(b) Gráfica de las coordenadas $y - z$.

Figura 3.17: Pruebas de las mediciones: 5 conjuntos de 30 mediciones cada uno, ambos ejes representan centímetros, estas mediciones no tienen interferencia.

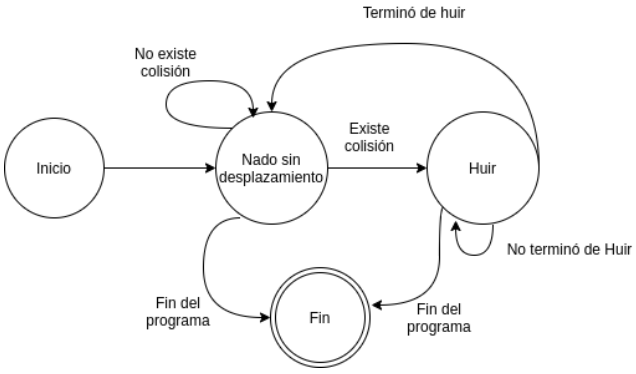


Figura 3.18: Diagrama de estados del comportamiento del pez.

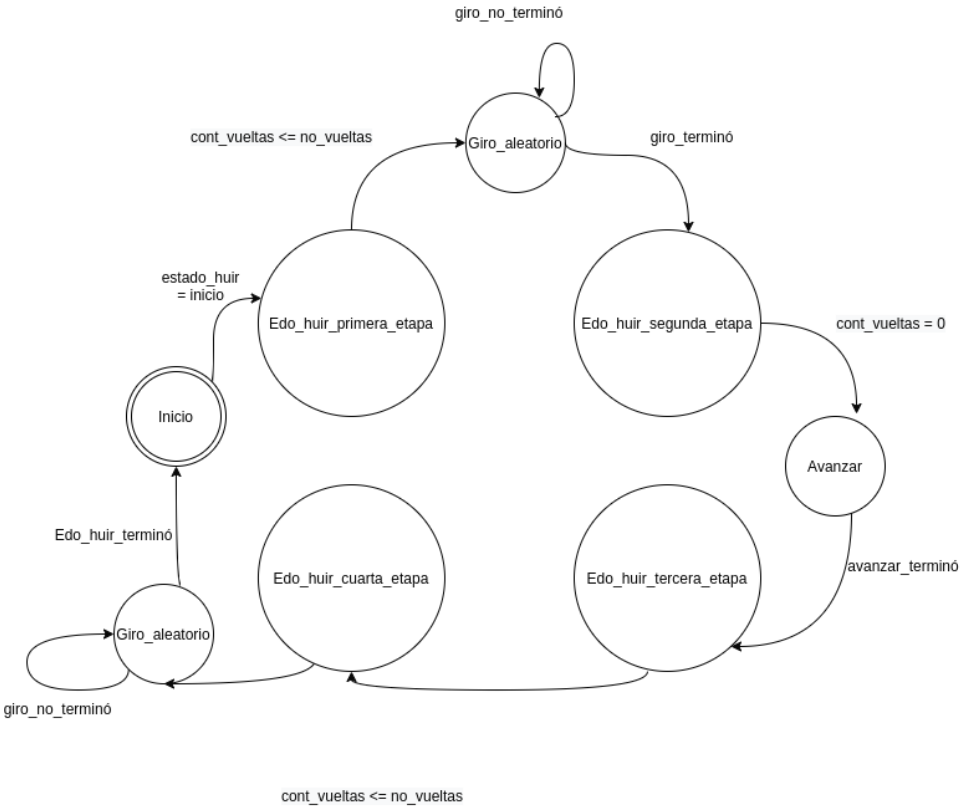


Figura 3.19: Diagrama de estados del comportamiento Huir.

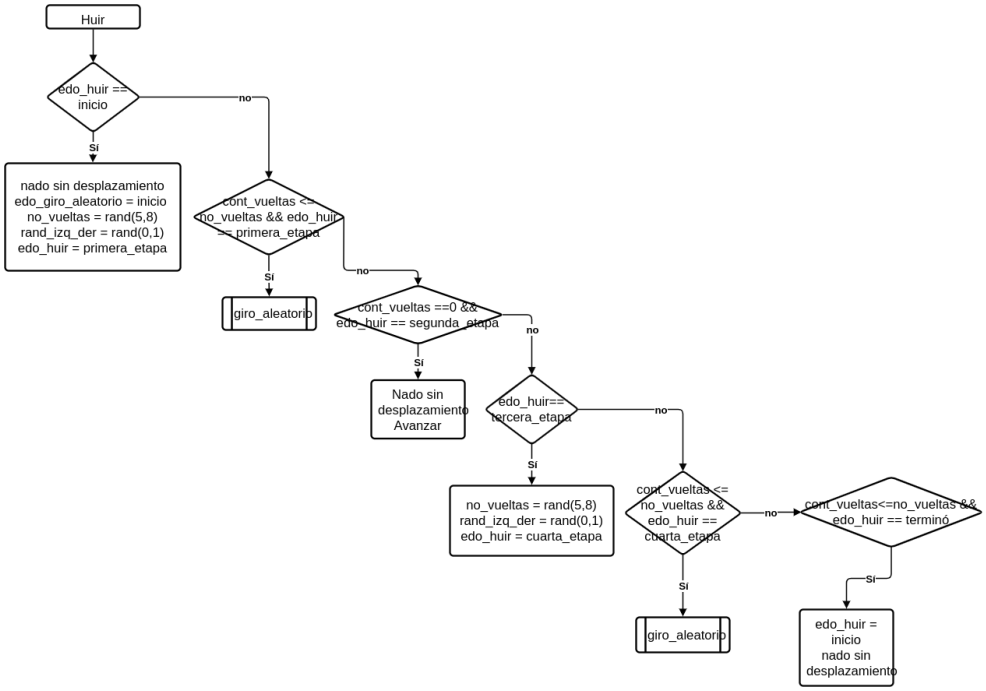


Figura 3.20: Diagrama de flujo : Huir.

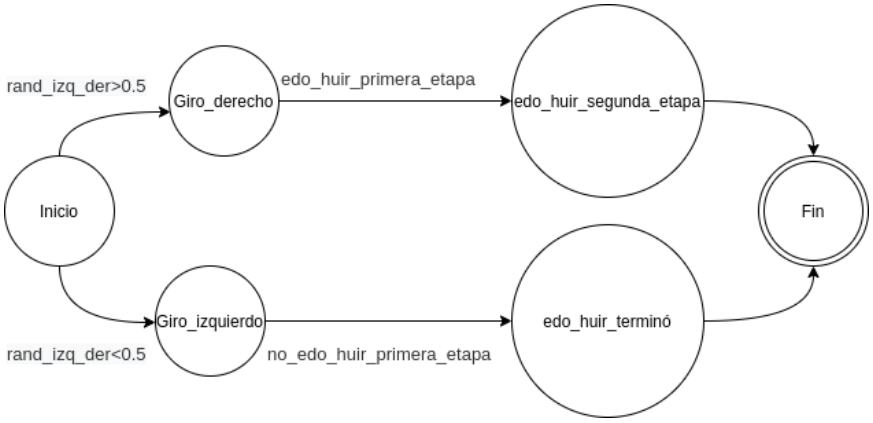


Figura 3.21: Diagrama de estados del giro aleatorio.

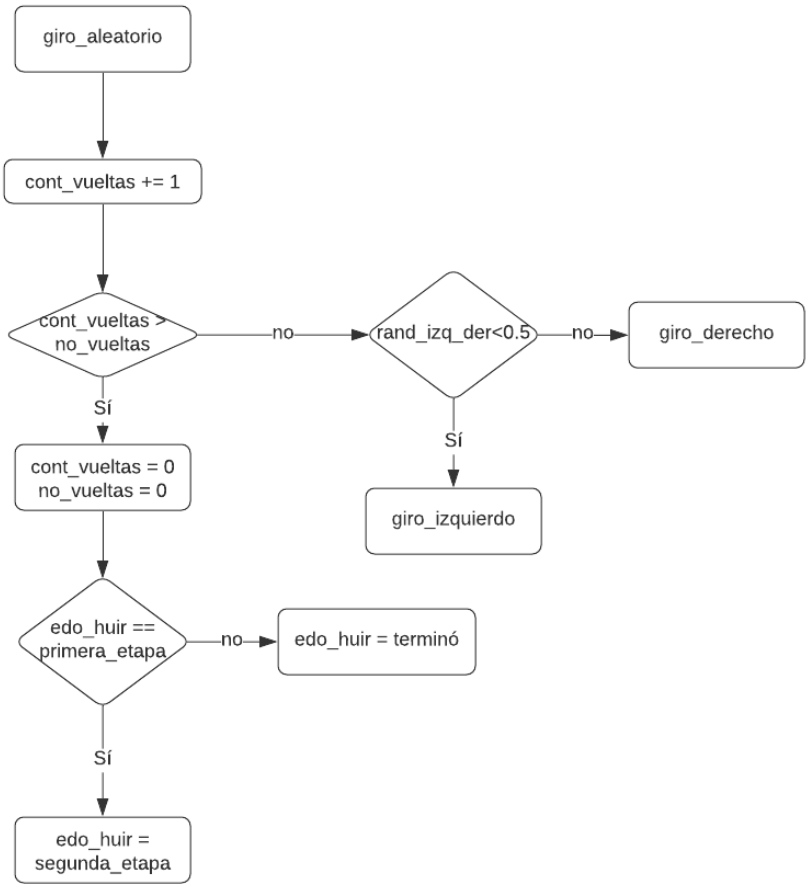


Figura 3.22: Diagrama de flujo : Giro aleatorio.

Capítulo 4

Metodología y validación

Este capítulo presenta la metodología de desarrollo y validación de todo el sistema realizado. El sistema fue dividido en cuatro módulos principales:

1. El reconocimiento de varios marcadores,
2. la visualización y animación del objeto virtual,
3. La obtención de las coordenadas del sensor, y
4. La integración y prueba de los tres módulos anteriores.

En cada sección de este capítulo se presenta cada módulo junto con las validaciones que se llevaron a cabo para asegurar el correcto funcionamiento. En forma general, el sistema fue construido de manera modular y jerárquica y se buscó siempre contar con una interfaz gráfica que permitiera verificar su funcionamiento correcto. De esta forma se pudo validar el sistema de manera paulatina a través de la interfaz gráfica, siendo la validación una serie de pruebas que de forma interactiva y visual hicieran corroborar su funcionamiento. A continuación se explicarán cada uno de los módulos.

4.1. Reconocimiento de varios marcadores

El reconocimiento de los marcadores, como se mencionó en el capítulo 2, en la página 20, utilizando técnicas de procesamiento de imagen, se detecta el cuadrado que forma un marcador y el triángulo dentro de él. Los puntos del marcador son las posiciones de los vértices del cuadrado y del triángulo. El punto más cercano al centro del marcador debe descartarse y así se tienen seis puntos. Con los valores de estas seis posiciones se calcula su matriz lambda asociada. En la tabla 4.1 se muestran las tres matrices lambda, en forma de una cadena concatenada, de los tres marcadores que se usaron en este trabajo (y que se muestran en la figura 2.6, en la página 16).

El diseño de los marcadores está basado en las métricas mencionadas en [14], de donde se están utilizando las medidas para marcadores de 6 puntos. A continuación se muestra en la tabla 4.1 los resultados de la matriz lambda para cada marcador

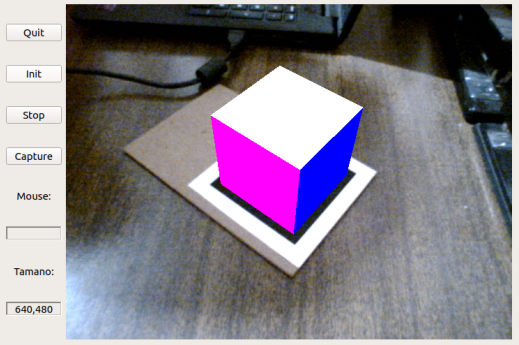
Etiqueta del marcador	Resultado matriz lambda	Perturbación Máxima
1	432102**1**43232214	12.020815
2	432101**2**43132214	8.753030
3	432101**3**42231234	16.263456

Cuadro 4.1: Tabla con los datos de los marcadores

y su perturación máxima. En negritas se encuentra el número en la posición 7 (de izquierda a derecha) de la matriz lambda. Ese número también sirve para identificar a cada marcador.

Se contaba con una versión de software que reconocía un solo marcador. Para agregar más, se utiliza la misma función de reconocimiento que devuelve el identificador y comparar este identificador asociado a cada marcador. La figura 4.1 muestra el reconocimiento de los nuevos marcadores, que se validó mostrando un objeto virtual sencillo para cada marcador. Para la interfaz creada, se cambia el marcador y se muestra un objeto virtual distinto (un cubo, o un cilindro sin tapas, o un tetraedro) sobre de él, tal como se muestra en la figura 4.1.

En el software inicial se buscaba un marcador 30 veces por segundo, esto es, en cada uno de los marcos de video. Este proceso hacia inestable la vista del objeto pues sí en un marco se había encontrado el marcador, pero no se reconocía en el siguiente, el objeto no se dibujaba y la visualización se generaba intermitente. Por ello se agregó la parte del seguimiento del marcador, en la cual una vez detectado este, el objeto virtual se muestra en un siguiente marco en la última posición donde se encontró el marcador hasta que este sea encontrado nuevamente. Esto hace que el objeto no desaparezca si eventualmente no se reconoce el marcador en alguno de los marcos. El procceso de validación fue hacer la prueba donde, una vez identificado el marcador y dibujado el objeto virtual, se quitó de la escena, de forma abrupta el marcador. En la versión anterior del software no se mostraría el objeto virtual si el marcador ya no aparece en la escena, pero en esta versión, con seguimiento, se puede observar que el objeto virtual sigue apareciendo en la pantalla. Esto es porque ha sido dibujado en la última posición donde se encontraba el marcador. La figura 4.2 muestra esta prueba.
En el reconocimiento del marcador uno de los pasos que se realiza es el de identificar las componentes conectadas. Este proceso se lleva a cabo a través de OpenCV con la función *connectedComponentsWithStats* [28], la cual devuelve una lista de las cajas que envuelven a cada componente y otra lista con las estadísticas (la posición del vértice superior y a la izquierda de la caja, el ancho y largo de la caja, y el área de la componente conectada). Primero se busca el marcador en cada una de las cajas encontradas en cada marco. El proceso de seguimiento toma en consideración buscar el marcador solo en las cajas que intersecan a la caja donde se encontró el marcador por primera vez. Esta es una de las mejoras que se hizo una vez se comenzó a probar el software con los nuevos marcadores, pues solo es indispensable buscar el marcador



(a) Imagen de un cubo en el marcador 1.

(b) Imagen de un cilindro en el marcador 2.

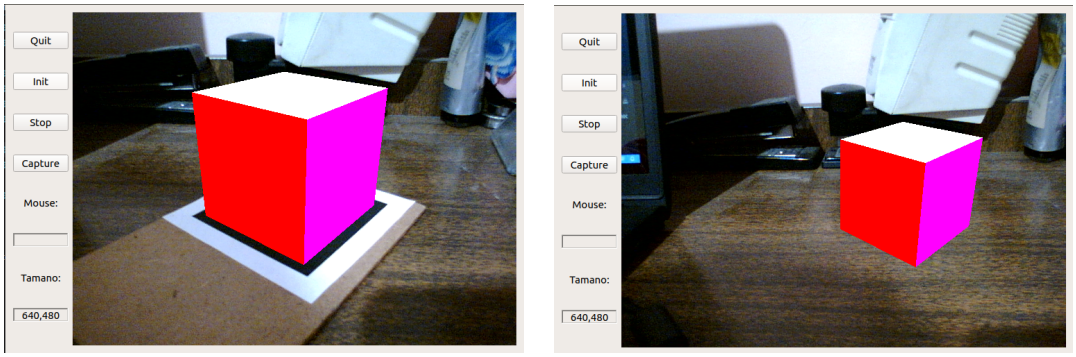
(c) Imagen de una piramide en el marcador 3.

Figura 4.1: Prueba del reconocimiento de nuevos marcadores con objetos virtuales sencillos.

en todas las componentes conectadas hasta la primera aparición de este, después de eso deberá estar en el mismo lugar o cerca, del que se encontró la primera vez. El diagrama de estados de la figura 4.3 muestra el proceso que se llevaba a cabo inicialmente, se muestra también un diagrama de flujo en la figura 4.4 con sentencias *if* de este proceso. En la figura 4.5 se presenta el diagrama de estados con seguimiento, mientras en la figura 4.6 se muestra un diagrama con sentencias *if* del proceso con seguimiento explicado en este párrafo.

4.2. Creación del objeto virtual y su animación

El primer objeto virtual modelado fue el pez naranja animado. Una vez realizado el modelo en Blender se debía exportar y encontrar la forma de visualizarlo a través de OpenGL. Para obtener los datos del modelo de Blender lo que se hizo fue exportarlo en formato obj. Este formato es un archivo de definición de geometría 3D [29][30], creado por *Wavefront Technologies* para su paquete de animación, el formato de archivo es abierto y se ha adoptado por muchas aplicaciones de gráficos 3D. El formato



(a) Imagen de un cubo en el marcador 1 reconocido.

(b) Imagen del cubo a pesar de no encontrarse el marcador en ese marco.

Figura 4.2: Prueba del proceso de seguimiento.

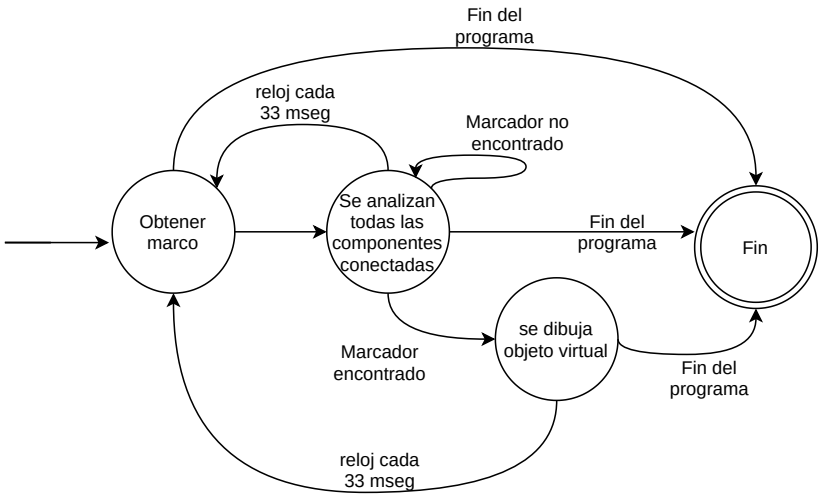


Figura 4.3: Diagrama de estados del proceso de búsqueda del marcador antes de implementar seguimiento.

obj es un formato de archivos simple que contiene las coordenadas de cada vértice, las posiciones u, v de cada coordenada de textura, la lista de normales de los vértices y la lista de caras formadas por los vértices.

Ya que se tenía el archivo obj exportado de Blender se buscó una biblioteca para cargar los datos del archivo en un programa en C++ y visualizarlo con OpenGl. Se encontraron varias formas de hacer la lectura del archivo obj [31, 32, 33, 34], pero debido a la facilidad con que se podía adaptar a lo que se requería para este proyecto se eligió la encontrada en [35]. Esta forma muestra como realizar la extracción de los datos y presenta una visualización del objeto utilizando la biblioteca GLFW [36] para OpenGL en el uso de ventanas y la tecnología de shaders [37] para el proceso

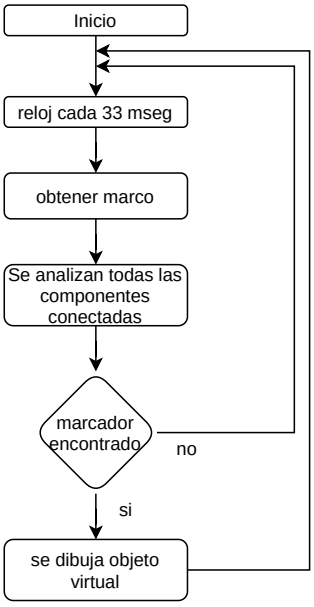


Figura 4.4: Diagrama de flujp del proceso de búsqueda del marcador antes de implementar seguimiento.

de texturizado del objeto. Se realizó una serie de primeras pruebas con este método sin hacer demasiadas modificaciones, de estas se obtuvieron los primeros detalles a modificar.

1. La escala en Blender es diferente al tamaño real o tamaño esperado, así que se buscó un método de equivalencia que dejara los datos en milímetros [38].

2. Hay que exportar el archivo obj de manera correcta, es indispensable seleccionar la casilla de *triangulate faces*, pues para realizar la visualización se indican conjuntos de tres vértices para cada cara. En la imagen 4.7 se muestran los paramétros correctos para exportar.

La imagen 4.8 muestra una de las primeras visualizaciones de la trucha con los datos obtenidos del archivo obj en una ventana que utiliza OpenGL.

Posteriormente se realizó un primer acercamiento a un prototipo de este módulo que pudiera ser integrado al sistema completo, para ello se buscó la manera de hacer la visualización en el entorno de Qt y no con la biblioteca GLFW. La documentación de Qt proporcionó un ejemplo de visualización de objetos con shaders[39], esto hizo que se tomara la decisión de utilizar el componente QWindow.
Se realizaron adecuaciones para visualizar un objeto con shaders en una ventana QWindow de Qt. El objeto tiene un modelo conocido del cual se tenían previamente los cálculos de su homografía. De esta forma se realizó la prueba colocando el objeto

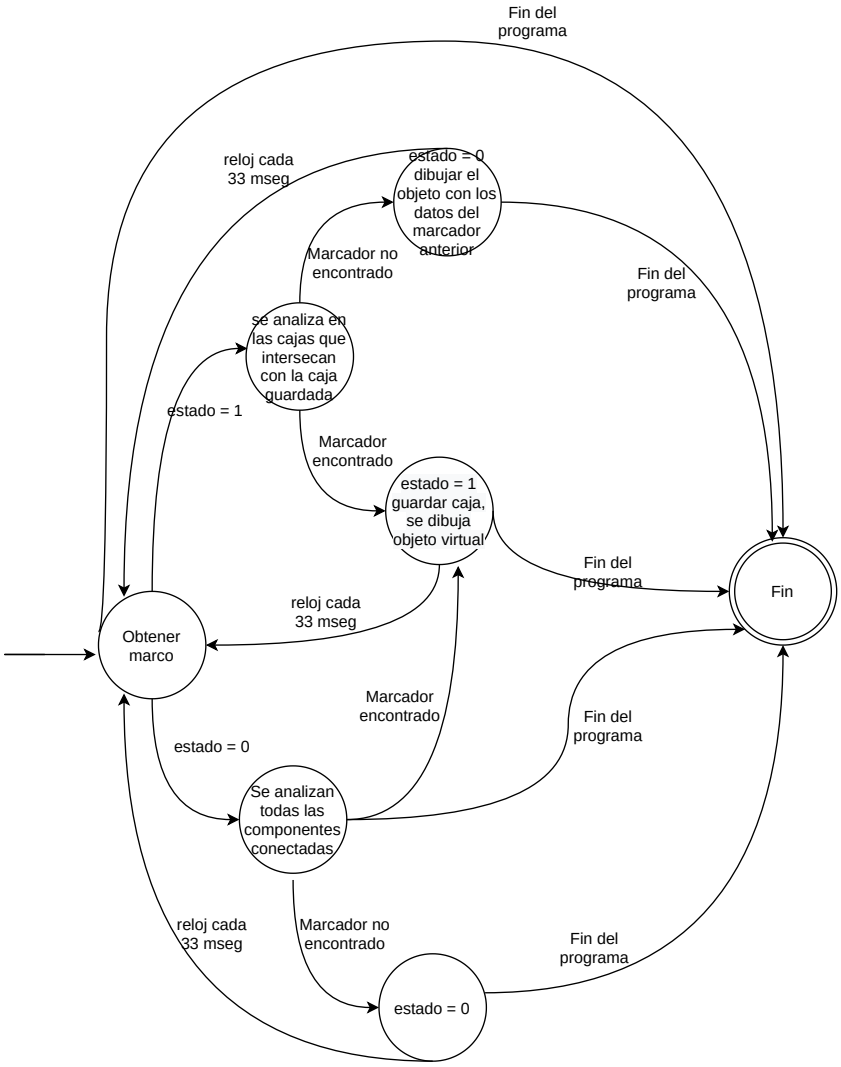


Figura 4.5: Diagrama de estados del proceso de búsqueda del marcador con seguimiento.

en pantalla y sobreponiendo los puntos de la homografía que fueron escritos en el código de forma estática. Esto para conocer el funcionamiento de QWindow con los shaders y poderlo integrar más adelante al software final. La imagen 4.9 muestra esta prueba. La imagen 4.10 muestra una prueba realizada usando textura con shaders en dos objetos.

Lo siguiente fue implementar la animación a la trucha ya mostrada, poniendo como fondo la imagen obtenida de la cámara. Para ello se utilizaron eventos de presionar teclas. La tecla I para iniciar la animación del nado sin desplazamiento, la S para detener la animación, R para realizar una rotación de 30^o.

Para implementar la animación se requiere guardar un arreglo con las posiciones

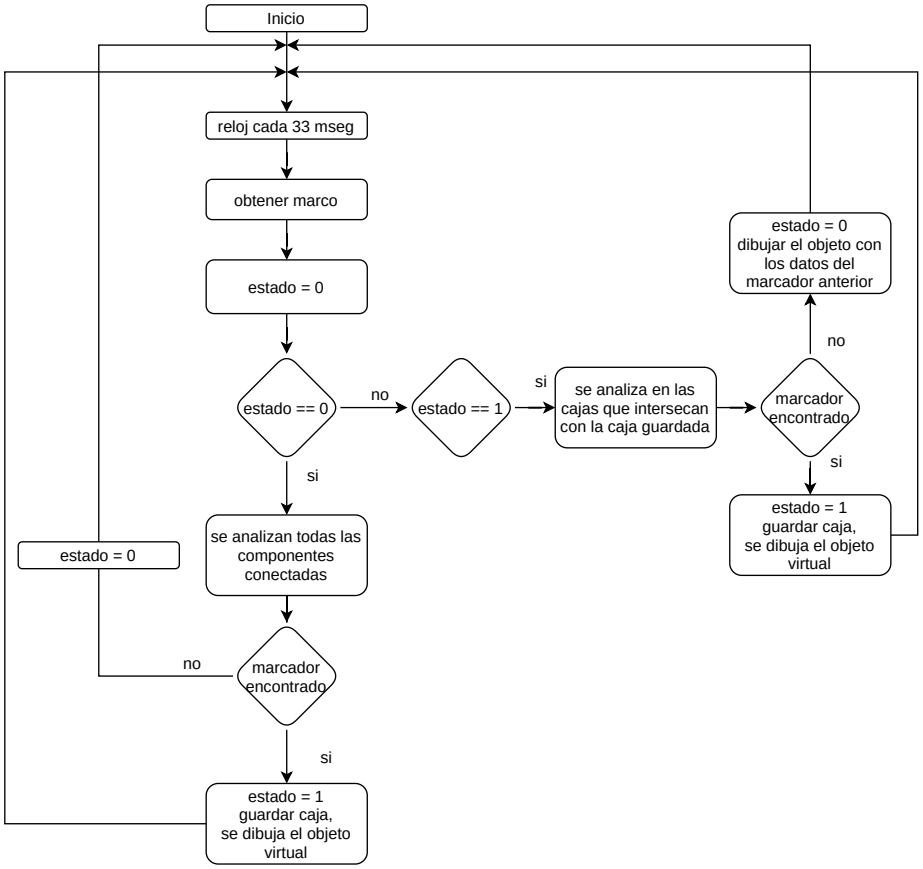


Figura 4.6: Diagrama de flujo del proceso de búsqueda del marcador con seguimiento.

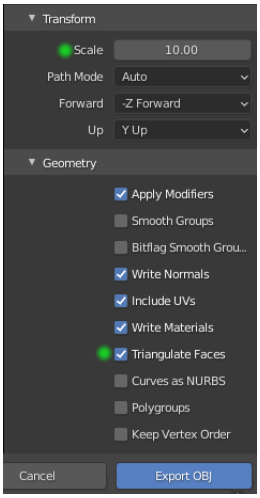


Figura 4.7: Paramétros para exportar el archivo obj.

originales de la trucha en el eje y, que es al que se le aplica la ecuación de onda senoidal. Lo que se hace es asignar una nueva y a cada uno de los vértices del obje-

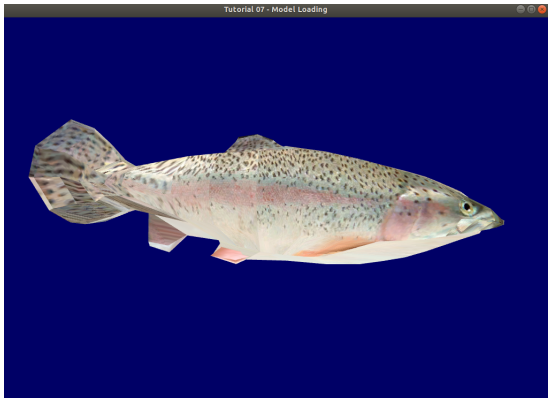


Figura 4.8: Visualización del objeto en una ventana con OpenGL.

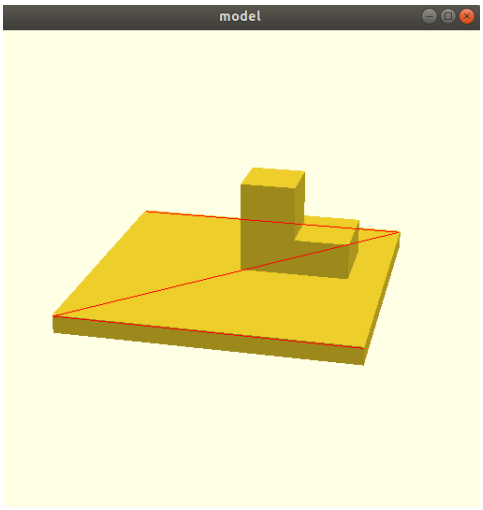


Figura 4.9: Modelo y puntos transformados en Qt con QWindow.

to aplicando la fórmula (3.1). La imagen 4.11 muestra un diagrama de flujo de este proceso.

4.3. Obtención de las coordenadas del sensor

Para el módulo de obtención de coordenadas del sensor se tenía el software mencionado en la sección 3.6.1, en la página 31, el cuál hace una petición de datos a través del comando "p" en intervalos de tiempo de 1 segundo. La parte importante de este módulo era identificar y conocer bien como entrega los datos el sensor, que tan precisos son y descubrir los casos en los que los datos se pudieran ver afectados. Las pruebas de precisión presentadas en el capítulo 3 sirvieron para cambiar el entorno de trabajo, la fuente magnética se fijó con un eje de coordenadas cartesianas de mano izquierda (igual que el de blender y el usado en OpenGL). Se tuvo que modificar el

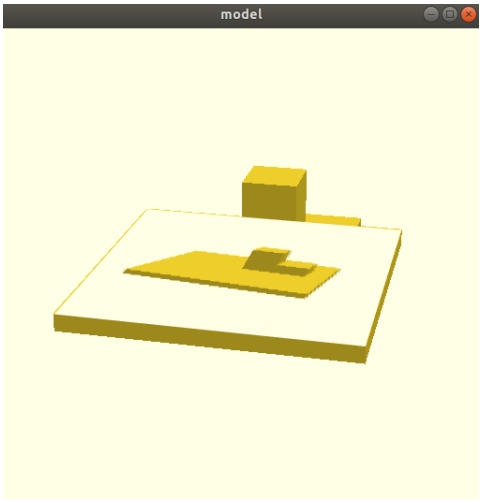


Figura 4.10: Dos objetos texturizados con shaders en Qt con QWindow.

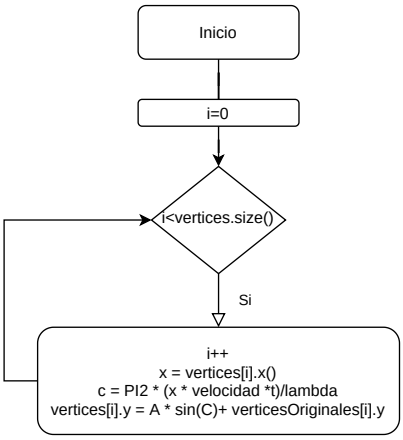


Figura 4.11: Diagrama de flujo de la implementación de la fórmula para la animación a los vértices del objeto.

dispositivo para fijar la distancia entre el sensor y el marcador, esto porque se encontró que cuando la pluma *Stylus* es movida hacia el eje "x" negativo de la fuente magnética los datos recibidos se vuelven inestables. La figura 4.12 muestra las medidas finales de este dispositivo.

Para la interacción era necesario conocer las medidas del centro del marcador, por lo cual se realizaron 10 mediciones, las cuales se promediaron para así dar las coordenadas a usar para la interacción con el objeto virtual. Estas coordenadas son las siguientes: 8.330596, 16.765722, 1.620367.

Además de poder hacer la petición haciendo uso del comando "p", el sensor Patriot cuenta con un "modo continuo" que entrega datos de forma ininterrumpida sin

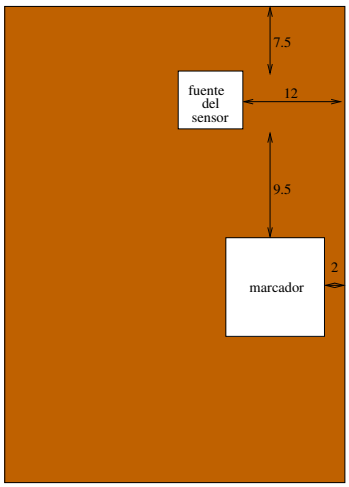


Figura 4.12: Dispositivo, versión 2, para fijar la distancia entre el sensor y el marcador, las distancias están dadas en cm.

necesidad de enviar ninguna petición. Se probó de este modo también, sin embargo, los resultados no fueron buenos. Las mediciones hacian cambios bruscos de signo y se perdían mediciones, por ello se optó por seguir haciendo la petición con el comando "p".

4.4. Integración del sistema

Ya que se tenían todos los módulos trabajados de manera independiente se realizó el proceso de integración. Hasta este punto se contaba con el software que reconocía los nuevos marcadores, este era visualizado en Qt utilizando para las ventanas el componente, QGLWidget. Como se mencionó en la sección anterior, el software anterior con el que se contaba usaba objetos QGLWidget, que permite el uso de muchos widgets para la construcción de la interfaz gráfica, y se cambió a un objeto QWindow, en el que aparece una sola ventana para la visualización. Se hizo una reestructuración completa del software que se tenía, para usarse del mismo modo que el software de visualización presentado en la sección 4.2, en la página 42. Se migró a este toda la parte del procesamiento de la imagen, el reconocimiento del marcador y el cálculo de la homografía.

La función de obtención de datos del archivo obj se modificó para hacer uso de objetos nativos de Qt tales como QVector3D para almacenar las coordenadas x, y, z de los vértices del objeto, y QPointF para almacenar las coordenadas uv_x y uv_y de los puntos de textura asociados al objeto. Además se omitió la parte de almacenamiento de las normales del archivo obj pues no se utilizan en esta tesis y obtenerlas era un trabajo innecesario para el procesador. Es importante mencionar que las pruebas estáticas de visualización, realizadas con una homografía precalculada, en la sección 4.2

sirvieron para identificar que el almacenamiento de las matrices de un objeto QMatriz (Qt) es contraria a la que se maneja en OpenGL y se se usaba en el software con QGLWidget.

Una matriz para QMatriz se inicializa de forma habitual como en C, renglón por reglón. En OpenGL las matrices se inicializan columna por columna. Debido a esto aunque la homografía era calculada de manera correcta al dibujar el objeto virtual este no se ajustaba al marcador, es decir no se veía como si perteneciera a la escena del mundo real y por el contrario solo se veia sobrepuesto. Se puede observar esta prueba en la imagen 4.13, al marcador se le pueden ver algunas lineas en rojo, lo cual indica que fue reconocido, y el objeto no estaba encima del marcador, como se puede apreciar en la imagen 4.13. Esto fue solucionado modificando la inicialización de la matriz de la homografía en el código.

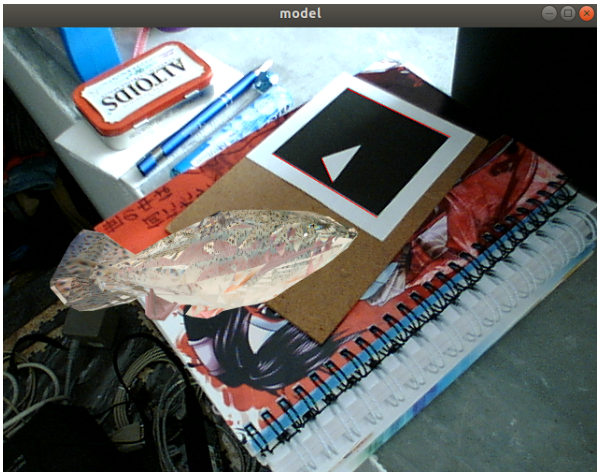


Figura 4.13: Objeto fuera del marcador, debido a la orientación de las matrices.

Hasta este punto se tenía la interfaz con QWindow, el proceso completo de identificación del marcador y el objeto virtual dibujado sobre el marcador en el mismo software, el siguiente paso fue integrar el módulo de sensado. La primera validación era mostrar los datos exactamente como los entrega el sensor, en consola, esta prueba no tuvo ningún inconveniente. Después se guardaron los datos sensados en un vector que serviría, más adelante, para la detección de colisiones, pero para este momento, permitió visualizar las coordenadas del sensor cuadro por cuadro. Se debe recordar que lo que permite visualizar la escena completa se está ejecutando cada 33 milisegundos, como se muestra en los diagramas de las figuras 4.4 y 4.6. La figura 4.14 muestra una línea roja que va de las coordenadas sensadas hacía el centro del marcador.

Lo siguiente fue implementar la detección de colisiones, para ello se tomó como referencia el centro promediado del marcador con el sensor, el cual fue mostrado en la sección 4.3 en la página 46, esto porque es en esta ubicación donde se dibuja el

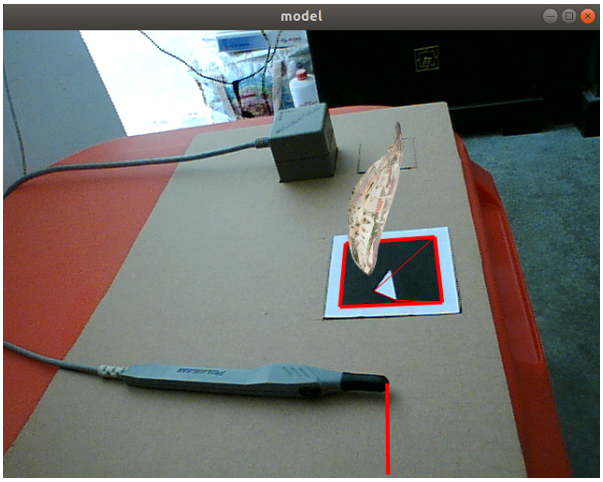


Figura 4.14: Prueba que muestra una línea a partir de las coordenadas obtenidas del sensor en dirección al centro del marcador.

objeto virtual. Posteriormente se delimitó una esfera alrededor del objeto virtual, la cual quedó con un radio de 7.8 cm. y una esfera de radio 1.5 cm. alrededor de la pluma "Stylus" del sensor, se muestra una representación gráfica en la figura 4.15.

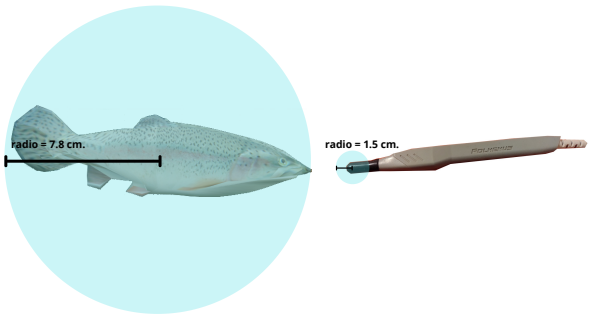


Figura 4.15: Dibujo de las esferas usadas para la detección de colisiones entre el pez y la punta del sensor.

Una vez con las esferas delimitadas se implementó la fórmula (2.7), de detección de colisiones, para validarlo, se hizo una prueba visual en la que cuando el sensor colisiona con la trucha la línea que se dibuja del sensor cambia de color rojo a verde. Como se muestra en la imagen 4.16.

Se implementarón los comportamientos de animación descritos en la sección 3.7, al realizar la pruebas se observó que la interacción no era fluida debido a que la forma

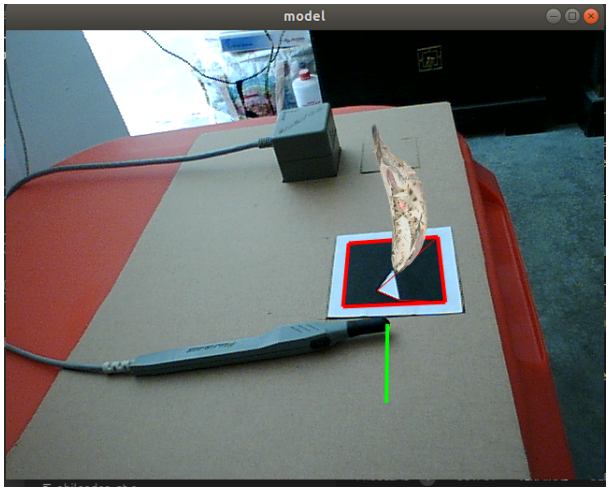


Figura 4.16: Colisión entre la pluma Stylus del sensor y la trucha.

en que se diseñó no permitía una nueva interacción hasta haber terminado el proceso completo de la interacción anterior, por lo cual no importaba si se acercaba el sensor o no, una vez iniciado un proceso de huida, pues las nuevas interacciones no tenian efecto alguno, es por ello que se relizó una modificación en la interacción que permitiera ver el proceso más fluido.

La nueva interacción verifica en todo momento el evento de colisión de la pluma *Stylus* del sensor con la esfera ubicada en el centro del marcador que cubre al objeto virtual cuando este se encuentra ahí. Si se detecta una colisión se comienza el proceso de huida, el cual incluye, igual que antes, realizar un giro aleatorio a la izquierda o a la derecha y el nado con desplazamiento que permite ver a la trucha lejos del centro del marcador, posteriormete se realiza otro giro para que la trucha se mantenga en un nado sin desplazamiento mirando hacia el marcador (hacia la cámara). Si el sensor no está haciendo colisión con la esfera del centro del marcador entonces la trucha vuelve a su posición inicial siguiendo los mismos pasos que realizó para huir de ahi. En esta nueva versión de interacción se redujo el ángulo de giro y la amplitud de la onda senoidal que se utilizaba para las vueltas a fin de que la animación no parezca cortada puesto que ya no se espera a los puntos donde las ondas se cruzan para poder hacer el cambio de la onda senoidal con la que se hace el giro a la onda senoidal con la que se hace el nado sin desplazamiento. En el diagrama des estados de la figura 4.17 se muestra este nuevo comportamiento. En la figura 4.18 se muestra el manejo de estados utilizando sentencias *if* para lograr este nuevo comportamiento de la interacción.

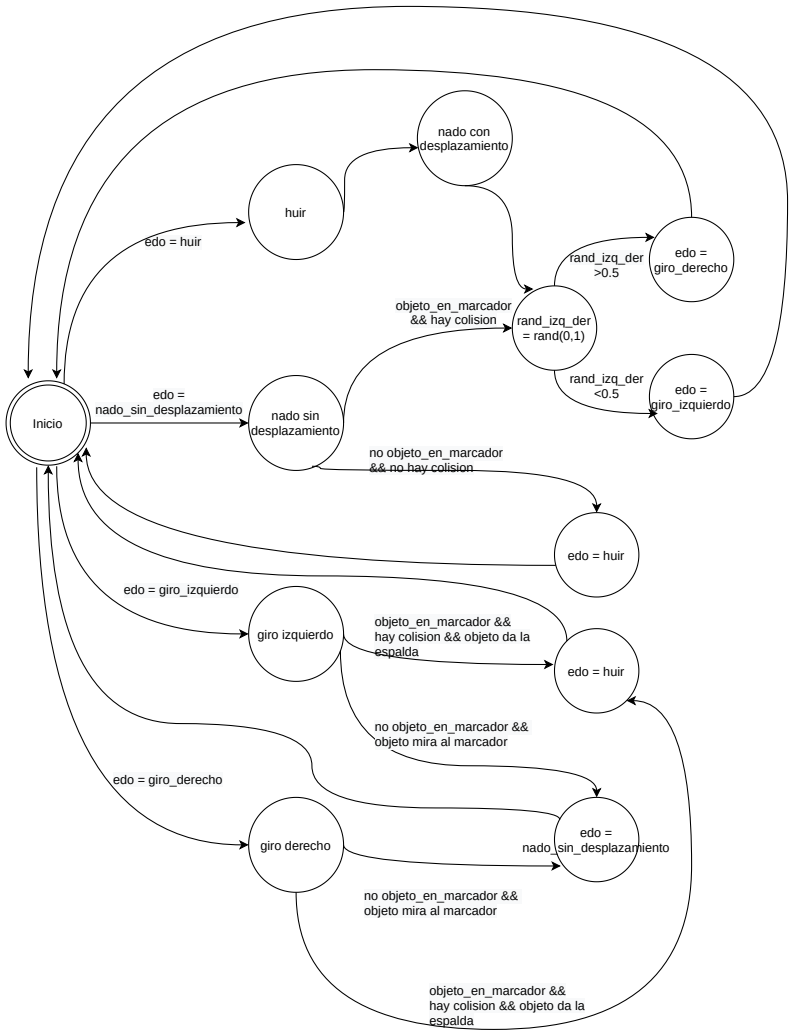


Figura 4.17: Diagrama de estados de la interacción del sensor con el objeto virtual.

4.5. Integración del objeto virtual Ave

A lo largo del desarrollo del sistema se identificaron las herramientas necesarias para implementar un ave como objeto virtual, en alguno de los marcadores disponibles reconocidos. Aunque se realizó también una investigación acerca de los modelos de animación exitentes para el vuelo de las aves, al hacer las simulaciones del nado del pez se encontró una similitud entre la onda senoidal usada para el nado y el aleteo de las alas de un ave, por ello se utilizó la onda senoidal con diferentes valores para realizar la simulación del vuelo del ave. En primer lugar se realizó el modelo del ave, para el cual se usó la imagen de una grulla. Esta imagen se puede ver en la figura 4.19.

Posteriormente se realizó una simulación para determinar los valores necesarios en la fórmula de la animación. Es importante recalcar que para este modelo se realiza un incremento en la amplitud de onda a fin de obtener una amplitud mayor conforme se

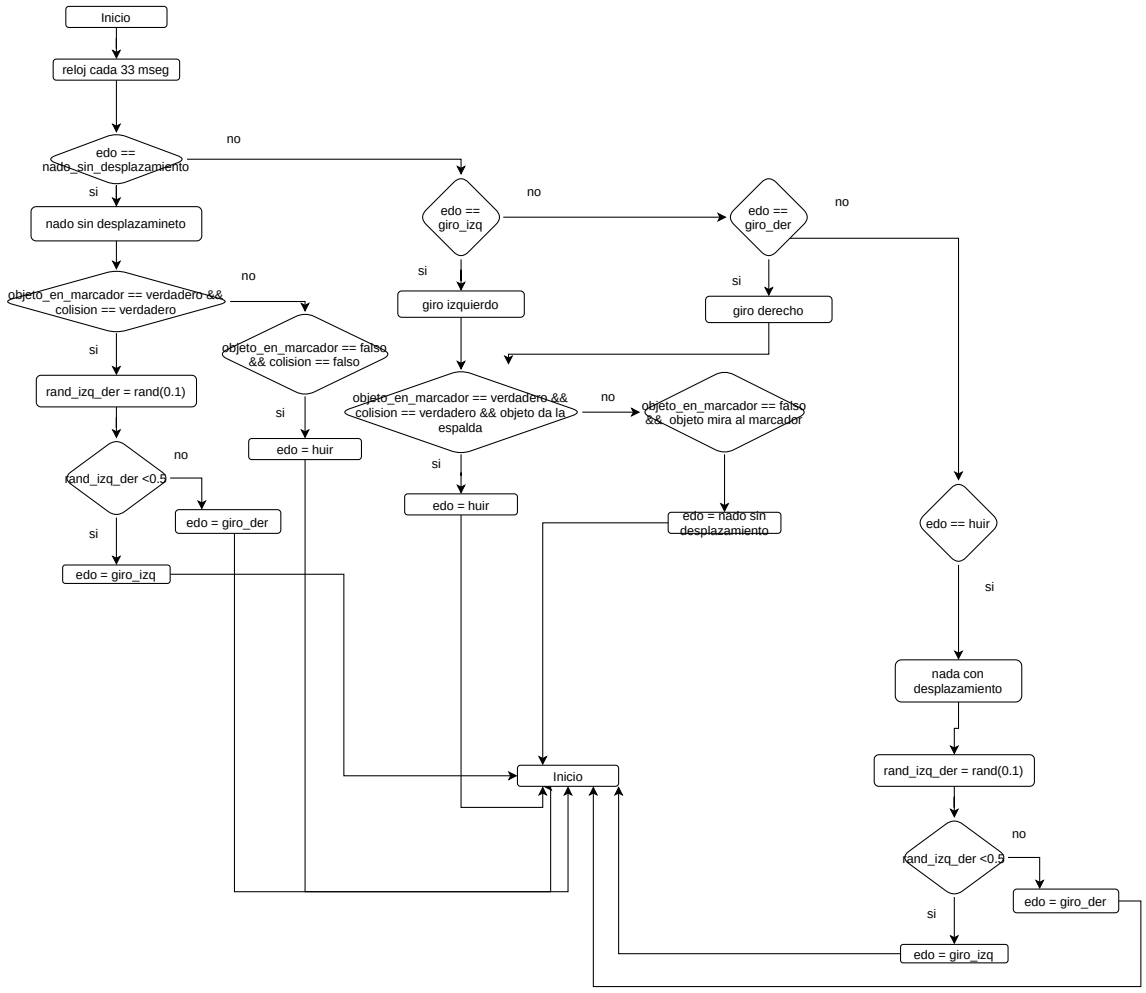
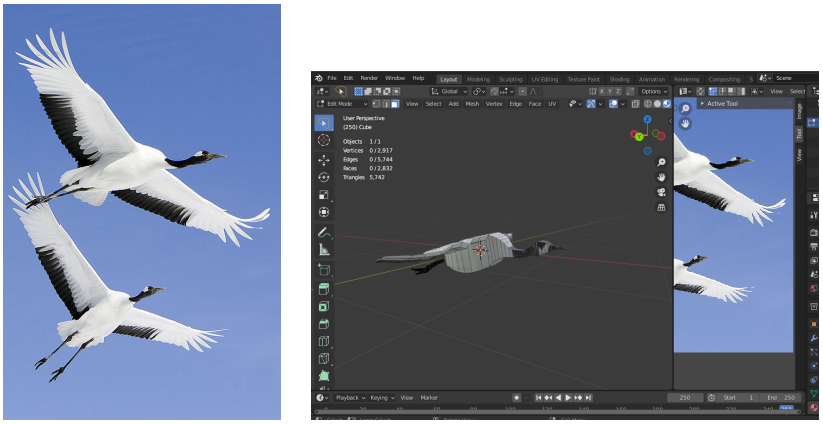


Figura 4.18: Diagrama de flujo de la interacción del sensor con el objeto virtual.

aleja el ala del cuerpo del ave, pues es donde se hace más grande esa apertura. Por lo tanto la amplitud de onda inicia en 1.55 y se multiplica por 1.12 en cada vértice más alejado. La velocidad de onda y el lambda es mucho mayor que en los modelos del pez pues no se requiere ver más de una cresta en cada ala por cuadro, por lo tanto de ocupa una velocidad de 390 y un lambda de 130. Se puede ver el resultado de estas simulaciones en la figura 4.20.

De estas primeras simulaciones se tomaron los datos que se necesitaban para el movimiento de un ala del ave, se procedió entonces a realizar una simulación donde se observara el comportamiento de las dos alas, para lo cual se dividió el largo de los puntos en dos partes que asemejaran las dos alas vistas de frente. Fue entonces que se encontro el problema de que la onda senoidal no es par, por lo que no se podía reproducir, tan fácil, el movimiento de las dos alas, se opto por utilizar la función coseno para no realizar más ajustes en las coordenadas del objeto. Además se identificó la mayor amplitud en el ala derecha para ponerla como amplitud inicial en el ala

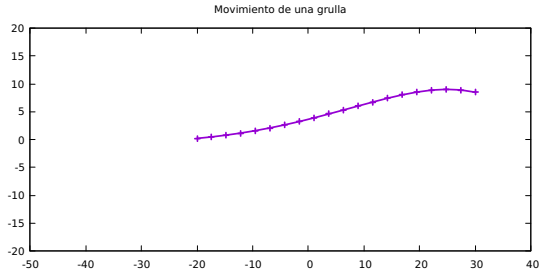


(a) Imagen de una grulla.

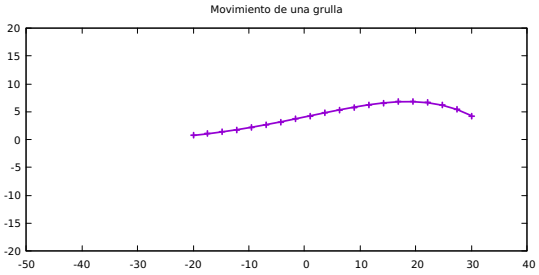
(b) Modelo virtual de una grulla.

Figura 4.19: Modelos de una grulla.

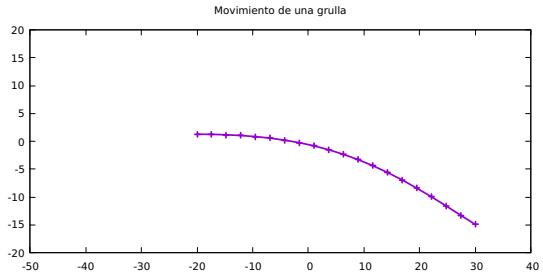
izquierda y en esta realizar una disminución de la amplitud en lugar de un incremento como en el caso del ala derecha. La figura 4.21 se muestra el tiempo estas simulaciones.



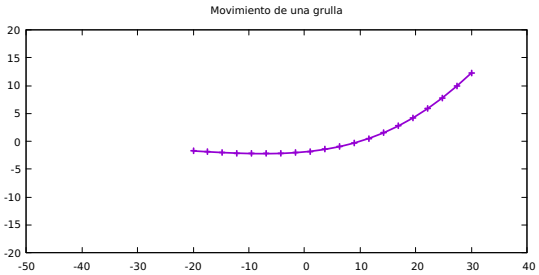
(a) Tiempo 0 de la simulación.



(b) Tiempo 7 de la simulación.

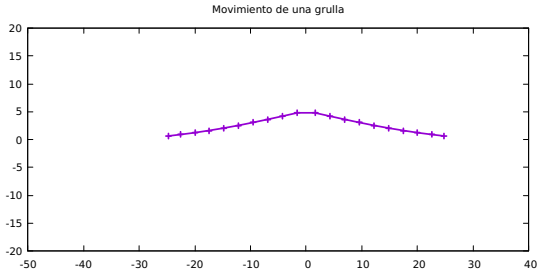


(c) Tiempo 14 de la simulación.

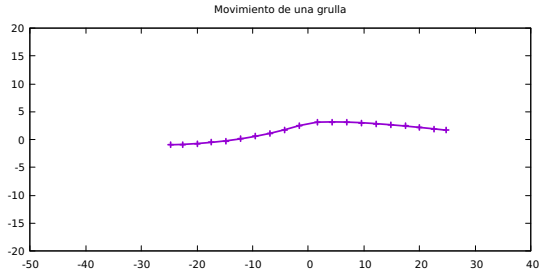


(d) Tiempo 21 de la simulación.

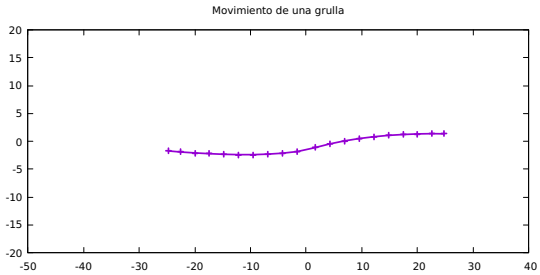
Figura 4.20: Simulación del movimiento del ala de una grulla.



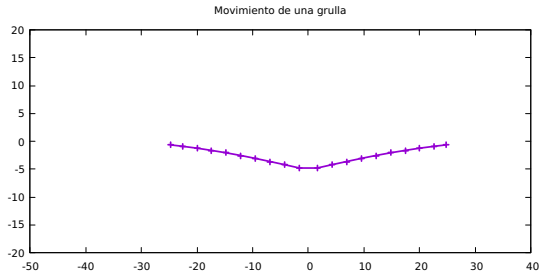
(a) Tiempo 0 de la simulación.



(b) Tiempo 7 de la simulación.



(c) Tiempo 14 de la simulación.



(d) Tiempo 21 de la simulación.

Figura 4.21: Simulación de las dos alas de una grulla.

Capítulo 5

Conclusiones

A lo largo de este trabajo se desarrollaron los objeivos planteados inicialmente, con lo que se obtuvieron los siguientes resultados:

Se desarrolló una interfaz de realidad aumentada que consiste en cuatro componentes principales: marcadores basados en tipo de orden, tres objetos virtuales articulados, un sensor de posición y una cámara de una playstation 3. La interfaz se desarrolló en Qt y C++, el mundo aumentado se visualiza en la pantalla de un monitor.

Los marcadores basados en tipo de orden permiten encontrar el emparejamiento de los puntos en la imagen del marcador y los puntos del modelo del marcador sin trasformar la imagen. La interfaz maneja tres marcadores distintos, uno por cada uno de los objetos virtuales diseñados. Estos objetos virtales fueron: una trucha, un pez animado y una grulla. Para animarlos se utilizó un modelo muy sencillo que usa la ecuación de onda, para el movimiento ondulatorio de los peces y el aleteo de la grulla. La forma de animar al pez o a la grulla es definiendo estados y comportamientos. El modelo programación de los estados y comportamientos, es un modelo tipo máquina de estados, lo cual permite hacer combinaciones de los estados para generar los comportamintos.

Con el sensor se puede interactuar con el objeto virtual que se dibuja encima del marcador. El marcador permite conocer su posición y orientación y como se usa una sola cámara esta ya está calibrada, o lo que es lo mismo, ya conocemos la transformación en perspectiva que introduce su uso.

La interacción del sensor con el objeto virtual se da cuando la pluma Stylus del sensor toca el objeto virtual, los peces o la grulla que en este caso se encuentran en un estado neutral en el que los peces están nadando sin moverse de su lugar al igual que el ave aletea sin cambiar su posición. Una vez que se hace contacto virtual con ellos a través del sensor, reaccionan huyendo de la posición donde se encuentra la pluma Stylus. Al quitar la pluma tanto los peces como el ave, regresan a su posición inicial para seguir nadando o aleteando.

El proceso para verificar el correcto funcionamiento fue por medio de la realización de pruebas prácticas en las que se usó la interfaz gráfica para observar que tanto la

aparición de los objetos dependiendo del marcador, así como la animación e interacción con ellos a tráves del sensor se ejecutarán de manera integral y fluida, dando al usuario la sensación de interacción con los peces y la grulla.

5.1. Trabajo a futuro

- Diseñar y animar más objetos virtuales y para poderlos usar todos al mismo tiempo sería necesario agregar más instancias del marcador.
- Desplegar e interactuar con varios objetos virtuales al mismo tiempo. En la interfaz actual solo se puede trabajar con un solo objeto virtual en un instante de tiempo.
- Mejorar la programación de los estados y comportamientos, que utiliza un modelo de máquinas de estado. Esto permitiría la programación más amigable de más estados y comportamientos.
- Agregar multiprocesamiento con varios hilos para la interacción con más objetos virtuales simultáneamente y con más sensores.
- Agregar más comportamientos al objeto virtual. Por ejemplo, para el ave, si la interacción se da desde arriba como si se tocara la cabeza del objeto, ésta deberia tomar un estado de reposo y dejar de volar para pararse en suelo.
- Interactuar de forma distinta, por ejemplo, hacer que si se toca el objeto desde arriba el programa se cambie a un modo de solo visualización del objeto y que con gestos (movimientos) del sensor ahora lo que se pueda hacer sea escalar, girar, o mover el objeto para su sola observación. Y si el objeto se toca de frente entonces se cambie el comportamiento normal de huida y regreso que tiene el objeto.

Bibliografía

[1] Ronald T. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 08 1997.

[2] Bernhard Preim and Charl Botha. Computer-assisted medical education. In *Visual Computing for Medicine*, pages e101–e169. Elsevier, 2014.

[3] T. R. Coles, D. Meglan, and N. W. John. The role of haptics in medical training simulators: A survey of the state of the art. *IEEE Transactions on Haptics*, 4(1):51–66, 2011.

[4] Miriam Mecate Zambrano. Interacción con objetos deformables. Master's thesis, CINVESTAV, 2008.

[5] Cristina Martin-Doñate. Interfaces hápticos. aplicación en entornos virtuales. *XVI Congreso Internacional de Ingeniería Gráfica*, pages 1–9, 06 2004.

[6] Samuel Thomas McJunkin. *Transparency improvement for haptic interfaces.* PhD thesis, Rice University, 2007.

[7] José Ribelles Claire Lastennet. Una aplicación informática para la enseñanza de las transformaciones geométricas 3D. Universitat Jaume I,, 2001-2002.

[8] M. Pauline Baker Donal Hearn. *Computer Graphics.* Prentice Hall, 1995.

[9] Andrew Zisserman Richard Hartley. Multiple view geometry in computer vision, 2000.

[10] Richard Szeliski. *Computer vision: algorithms and applications.* Springer Science & Business Media, 2010.

[11] Luis Gerardo De la Fraga. Apuntes de clase. El método de Zhang para autocalibrar una cámara. Apuntes de clase. El método de Zhang para autocalibrar una cámara, May 2020. Apuntes de la clase.El método de Zhang para autocalibrar una cámara.

[12] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:1330 – 1334, 12 2000.

[13] Heriberto Cruz Hernández and Luis Gerardo de la Fraga. Order type dataset analysis for fiducial markers. *Data in Brief*, 20:1068–1072, 2018.

[14] Luis Gerardo de la Fraga and Heriberto Cruz Hernández. Optimizing the maximal perturbation in point sets while preserving the order type. *Mathematical and Computational Applications*, 24(4), 2019.

[15] María Luisa Pinto-Salamanca, Jorge Iván Sofrony-Esmeral, and Daniel Fernando Jiménez. Detección de colisiones con librerías v-collide y physx para interacción virtual con interfaces hápticas. *Revista de Investigación, Desarrollo e Innovación*, 5(2):119–128, 2015.

[16] Blender, software libre para modelado 3D. https://blender.org. Checado por última vez el 24 sep, 2021.

[17] Blender 3D: Noob to pro. https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro. Checado por última vez el 24 sep, 2021.

[18] Georgina Priego. *Modelado de pez en blender*, 2020. https://www.youtube.com/watch?v=cbJfNFXaUPo.

[19] Santiago. *Cinematix, Blender, Introducción a las texturas*, 2020. https://www.youtube.com/watch?v=JN6OZLVNRpk&t=555s.

[20] Laura Hernández Rielo. Animación procedimental: Dando vida a través del código, 2019.

[21] Inmaculada Coma Marcos Fernández. Animación 3D, 2008.

[22] C. Kim, M. Shin, S. Jeong, I. Kang, E. Kim, and B. Kim. Real-time motion simulation of artificial fish for virtual marine world. In *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*, pages 253–260, 2007.

[23] Innovation in Motion Polhemus. Patriot, two-sensor, 6dof tracker that dlivers unmatched value, 2017. https://polhemus.com/_assets/img/PATRIOT_brochure.pdf.

[24] Innovation in Motion Polhemus. Patriot, user manual, 2013. https://polhemus.com/_assets/img/PATRIOT_User_Manual_URM03PH170-J.pdf.

[25] ubuntu manuals. Termios, 2019. http://manpages.ubuntu.com/manpages/bionic/es/man3/termios.3.html.

[26] GNU manual. 17.4 terminal modes. http://www.gnu.org/software/libc/manual/html_node/Terminal-Modes.html.

[27] Geoffrey Hunter. Linux serial ports using C/C++, 2020. https://blog.mbedded.ninja/programming/operating-systems/linux/linux-serial-ports-using-c-cpp/.

[28] OpenCV. Structural analysis and shape descriptors. https://docs.opencv.org/4.5.3/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f.

[29] Wavefront .obj file. https://en.wikipedia.org/wiki/Wavefront_.obj_file#Vertex_indices. Checado por última vez el 27 sep, 2021.

[30] Pyeong-ho Choi Geon-hee Lee. A study on the performance comparison of 3D file formats on the web. *International journal of advanced smart convergence*, 8:65 – 74, 03 2019.

[31] Bly7-obj-loader. https://github.com/Bly7/OBJ-Loader. Checado por última vez el 27 sep, 2021.

[32] Cargar modelo 3D formato obj. http://acodigo.blogspot.com/2016/04/cargar-modelo-3d-formato-obj.html. Checado por última vez el 27 sep, 2021.

[33] protochron-objload. https://github.com/protochron/OBJLoad. Checado por última vez el 27 sep, 2021.

[34] tinyobjloader-tinyobjloader. https://github.com/tinyobjloader/tinyobjloader. Checado por última vez el 27 sep, 2021.

[35] Tutorial 7 : Model loading. http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/. Checado por última vez el 27 sep, 2021.

[36] Glfw. https://www.glfw.org/. Checado por última vez el 27 sep, 2021.

[37] Shaders. https://learnopengl.com/Getting-started/Shaders. Checado por última vez el 27 sep, 2021.

[38] 3Dimentional. Tabla de escalas para exportar modelos de blender 2.8 a cura, 2020. https://www.3dimentional.net/tabla-de-escalas-para-exportar-modelos-de-blender-2-8-a-cura/.

[39] Qt Documentation. Opengl window example, 2021. https://doc.qt.io/qt-5/qtgui-openglwindow-example.html.