

**Título:**

# Sistema de entrenamiento con Kinect



**Autor:** Ángel Rodríguez Estévez

**Fecha:** 19-06-2013

**Director:** Pau Fonseca i Casas

**Departamento:** Estadística e Investigación Operativa (EIO)

**Titulación:** Ingeniería Superior en Informática

**Centro:** Facultad de informática en Barcelona (FIB)

**Universidad:** Universidad Politécnica de Barcelona (UPC)



# Índice

1. Introducción.....	2
1.1 Presentación .....	2
1.2 Descripción del proyecto .....	2
1.3 Motivación .....	3
1.4 Objetivos .....	3
2. State of the Art .....	5
2.1 Kinect y el deporte.....	6
3. Dispositivos y Herramientas .....	8
3.1 Kinect .....	8
3.2 Desarrollar para kinect.....	11
3.2.1 Alternativas y elección.....	11
3.2.2 SDK oficial de Microsoft .....	13
3.2.3 KinectStudio.....	19
3.3 Sistema operativo .....	20
3.4 Lenguaje de programación .....	20
3.5 Windows Presentation Foundation (WPF) .....	20
3.5.1 Otra alternativa .....	21
4. Especificación.....	22
4.1 Conceptos relevantes para el desarrollo del proyecto .....	22
4.2 Definición de alcance .....	23
4.3 Características de KataCreator.....	23
4.4 Características de KataTraining.....	24
4.5 Actores .....	25
4.6 Decisiones preliminares .....	25
5. Desarrollo.....	26
5.1 Metodología.....	26
5.2 Borradores de vistas .....	26
5.2.1 Vista de la aplicación KataCreator.....	27
5.2.2 Vistas de KataTraining .....	28

5.3 Casos de uso.....	31
5.3.1 Diagrama interacción Usuario Sistema de KataCreator:.....	31
5.3.2 Diagrama interacción Usuario Sistema de KataTraining:.....	36
5.4 Descripción de clases .....	39
5.4.1 Clases del modelo que comparten las dos aplicaciones .....	39
5.4.2 Clases de KataCreator.....	40
5.4.3 Clases de KataTraining.....	41
5.5 Pruebas y resultados.....	43
5.5.1 Participantes.....	43
5.5.2 Descripción general de las pruebas.....	44
5.5.3 Consideraciones y restricciones previas.....	45
5.5.4 Resultados de las pruebas .....	47
6. Planificación y presupuesto.....	49
6.1 Planificación .....	49
6.2 Presupuesto .....	52
6.2.1 Coste de hardware .....	52
6.2.2 Coste de software.....	52
6.2.3 Coste del personal .....	53
6.2.4 Coste total .....	53
7. Conclusiones .....	54
7.1 Revisión de objetivos y conclusiones generales .....	54
7.2 Trabajo futuro .....	55
8. Manual de usuario .....	57
8.1 Requisitos del sistema .....	57
8.2 Manual de KataCreator.....	57
8.2.1 Instalación.....	57
8.2.2 Abrir/ejecutar KataCreator.....	60
8.2.3 Ventana principal de KataCreator .....	61
8.2.4 Observaciones y recomendaciones para la creación de una kata .....	68
8.2.5 Desinstalación de KataCreator .....	70
8.3 Manual de KataTraining.....	70

8.3.1 Instalación.....	70
8.3.2 Abrir/ejecutar KataTraining .....	73
8.3.3 Ventana principal de KataTraining .....	74
8.3.4 Ventana de Visualización de una kata .....	76
8.3.5 Ventana de Entrenamiento de una kata .....	77
8.3.6 Desinstalación de KataTraining .....	80
9. Bibliografía.....	81

# 1. Introducción

## 1.1 Presentación

Este documento, realizado por Ángel Rodríguez Estévez, contiene la memoria del proyecto final de carrera “Sistema de entrenamiento con Kinect” para la titulación Ingeniería Informática de la Facultad de Informática de Barcelona (FIB). El proyecto ha sido dirigido por Pau Fonseca i Casas, profesor del departamento de Estadística e Investigación Operativa.

El proyecto corresponde a la modalidad del tipo A, un proyecto propuesto por el tutor del proyecto.

## 1.2 Descripción del proyecto

El proyecto tiene como objetivo desarrollar un sistema de entrenamiento a partir de los datos que podemos obtener de una **Kinect**. Concretamente, será un sistema de entrenamiento para **karate** y la Kinect nos permitirá el reconocimiento de movimientos básicos y **katas** (en karate una kata es una secuencia establecida de golpes y bloqueos lanzados al aire).

Este sistema pretende ser una herramienta para que aquellas personas que practican Karate puedan entrenar determinadas katas usando tecnologías actuales (la Kinect en este caso), así como iniciar a las personas interesadas en este deporte.

Para realizar dicho objetivo se realizarán dos aplicaciones:

- **KataCreator:** aplicación que se encargará de la captación de la secuencia de posturas que forman una Kata. Permitirá guardar una Kata con toda la información relevante. Será una herramienta para la creación de Katas que posteriormente se podrán practicar con *KataTraining*.
- **KataTraining:** aplicación que permitirá al usuario escoger una Kata de la lista de Katas disponibles para visualizarla o entrenarla. Esta será la aplicación principal destinada a las personas que quieran entrenar.

Se ha decidido realizar estas aplicaciones por separado por una decisión de diseño, las Katas tienen que tener un nivel de calidad para una experiencia satisfactoria por parte del usuario final. La idea es que sea gente preparada la que cree las Katas para entrenar.

## 1.3 Motivación

La tecnología cada vez está más integrada en nuestra sociedad, prueba de ello son los deportes. Los avances de la tecnología han creado nuevas oportunidades en la investigación deportiva. Ahora es posible analizar aspectos del deporte que antes se encontraban fuera del alcance de nuestra comprensión. Técnicas como la captura de movimientos o las simulaciones por ordenador han incrementado el conocimiento acerca de las acciones de los atletas y el modo en que estas pueden mejorarse. Las mejoras en tecnología también han servido para mejorar los sistemas de entrenamiento, en ocasiones asistidas por máquinas diseñadas para tal efecto.

Este proyecto pretende crear uno de estos sistemas de entrenamiento, usando la captura de movimientos a través de la Kinect (tecnología al alcance de cualquier persona), para un deporte en el que los movimientos y posturas son una parte esencial como es el **Karate**.

En el karate uno de los elementos más importantes son las katas, sucesión de técnicas de bloqueo y golpeo (posturas) que se ejecutan al aire contra adversarios imaginarios. Es una de las principales herramientas de entrenamiento en Karate. En Karate, para ascender de cinturón (categoría) es común que exijan el conocimiento y la puesta en práctica de determinadas Katas acordes al cinturón. ¿Por qué no ofrecer entonces un sistema de entrenamiento donde se puedan **visualizar** y **entrenar** una serie de Katas?

Hay que tener claro que un sistema de entrenamiento de katas no tiene como objetivo ser un sustituto del entrenamiento tradicional, sino un complemento para gente que ya realice este deporte.

## 1.4 Objetivos

En este apartado se listan los principales objetivos que se persiguen con la realización de este proyecto, no sólo a nivel de la aplicación sino también a nivel personal.

- Aprendizaje y uso del lenguaje de programación C#.
- Aprendizaje y uso de WPF (Windows Presentation Foundation).
- Aprendizaje y uso del SDK oficial de Kinect.
- Aprendizaje de diferentes herramientas que vienen con el SDK de Kinect, como por ejemplo KinectStudio.
- Desarrollo de aplicación para captación de posturas de katas (KataCreator).
- Algoritmo de reconocimiento de posturas.

- Desarrollo de aplicación para visualización y entrenamiento de katas (KataTraining).

Se han barajado e investigado otros objetivos para este proyecto, como por ejemplo el uso de dos o más kinects para una mejor captación de las posturas, que finalmente no se han incluido dentro del alcance del proyecto.

## 2. State of the Art

La captura de movimientos (o motion capture [1]) es el proceso de grabación del movimiento de personas u objetos. Hoy en día es una tecnología muy utilizada y con diferentes aplicaciones prácticas como por ejemplo el cine, la medicina y el deporte entre muchos otros.



Figura 2.1: Captura de movimiento aplicado al golf.

Hablando de su uso histórico, si entendemos el motion capture como la captura de movimiento con dispositivos eléctricos no es hasta finales de 1800 cuando se empieza a trabajar en ello, en Inglaterra. Contrataron a Muybridge para demostrar que un caballo cuando corre nunca tiene las cuatro patas en el aire al mismo tiempo. Para ello desarrolló el zoopraxiscopio [2], una serie de cámaras activadas por el galope.

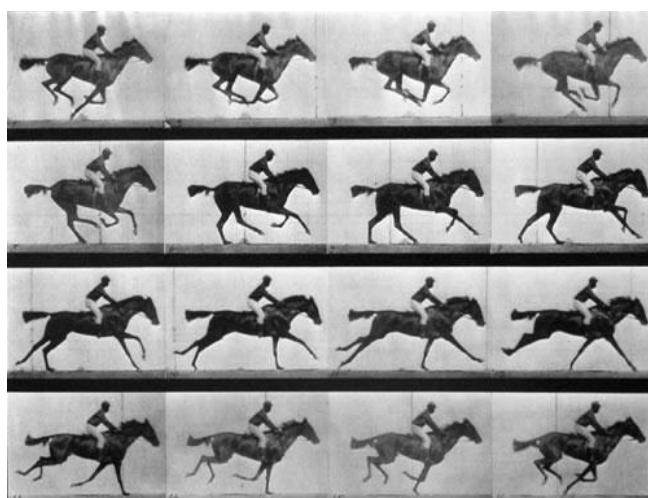


Figura 2.2: Capturas con zoopraxiscopio, primer dispositivo de motion capture.

Después de eso fueron llegando otros dispositivos (como la rotoscopia [3] por ejemplo) hasta llegar a sistemas mucho más complejos como los sistemas de captura de movimientos ópticos, mecánicos, etc. Estos son sistemas caros que no están al alcance de todas las personas, aun así se usan mucho en la actualidad (por ejemplo en la famosa película Avatar).



**Figura 2.3:** Sistema de captura de movimientos óptico.

En estos últimos años han aparecido dispositivos mucho más económicos y fáciles de usar para la captura de movimientos, como PS Move (Sony) o la Kinect (Microsoft). La tendencia imagino que seguirá estos pasos (al alcance de todas las personas), de momento ya ha sido anunciada la Kinect 2.

## 2.1 Kinect y el deporte.

La aparición de Kinect en noviembre de 2010 dejó entrever una serie de aplicaciones relacionadas con el deporte gracias a la captura de movimientos, prueba de ello es que el mismo mes que se puso a la venta la Kinect ya aparecieron aplicaciones de fitness, compitiendo de tú a tú con juegos de mucho renombre.



**Figura 2.4:** Juego que apareció al mismo tiempo que Kinect para clases de fitness.

En la actualidad hay una gran cantidad de estas aplicaciones para diferentes deportes, algunos de entretenimiento puro y otros más rigurosos. De aplicaciones más rigurosas podemos poner como ejemplo el UFC Personal Training (2011) en los que se usan técnicas y movimientos de artes marciales mixtas como parte de un programa de educación física, en el que expertos entrenadores de UFC guían al usuario.



Figura 2.5: Aplicación para entrenar UFC con Kinect.



Figura 2.6: Aplicación “Nike+Kinect Training” para ejercicios básicos con Kinect.

En el futuro seguirán apareciendo aplicaciones de este estilo con dispositivos cada vez mejores para ello. Ya han sido anunciados diversos dispositivos para la captura de movimientos que se espera que suplan las limitaciones de los dispositivos actuales.

# 3. Dispositivos y Herramientas

En este apartado se describen los dispositivos y herramientas relevantes para el desarrollo del proyecto. Se explican las características más importantes así como el motivo de su elección sobre otras alternativas.

## 3.1 Kinect



Figura 3.1: Microsoft Kinect.

Kinect [4] es un dispositivo creado por Alex Kipman, desarrollado por Microsoft para Xbox 360 que permite a los usuarios controlar e interactuar con la consola a través de gestos corporales y la voz. Fue lanzado en Norteamérica en 4 de noviembre de 2010 y en Europa el 10 de noviembre de 2010. En febrero de 2012, y después de darse cuenta del gran interés que había despertado su novedoso dispositivo, Microsoft decidió ofrecer a los desarrolladores “Kinect for Windows”, la versión oficial de Kinect para PC.

La principal diferencia entre la versión de Xbox 360 y PC es el Near Mode. El sensor de PC posee firmware que permite trabajar identificando cuerpos a partir de los 40 cms en lugar de los 80 cms de la versión de Xbox 360.

**Nota:** el desarrollo de la aplicación se ha realizado con la versión de Xbox 360 sin ningún problema, aunque el SDK de Microsoft da soporte únicamente para la versión de PC.

### 3.1.1 Componentes



Figura 3.2: Componentes de Kinect.

#### 1. Sensores de profundidad.

Son el elemento clave del funcionamiento de la Kinect, y son los encargados de hacer el seguimiento del cuerpo de los jugadores. Se basan en dos cámaras de infrarrojos que construyen un mapa de profundidad.

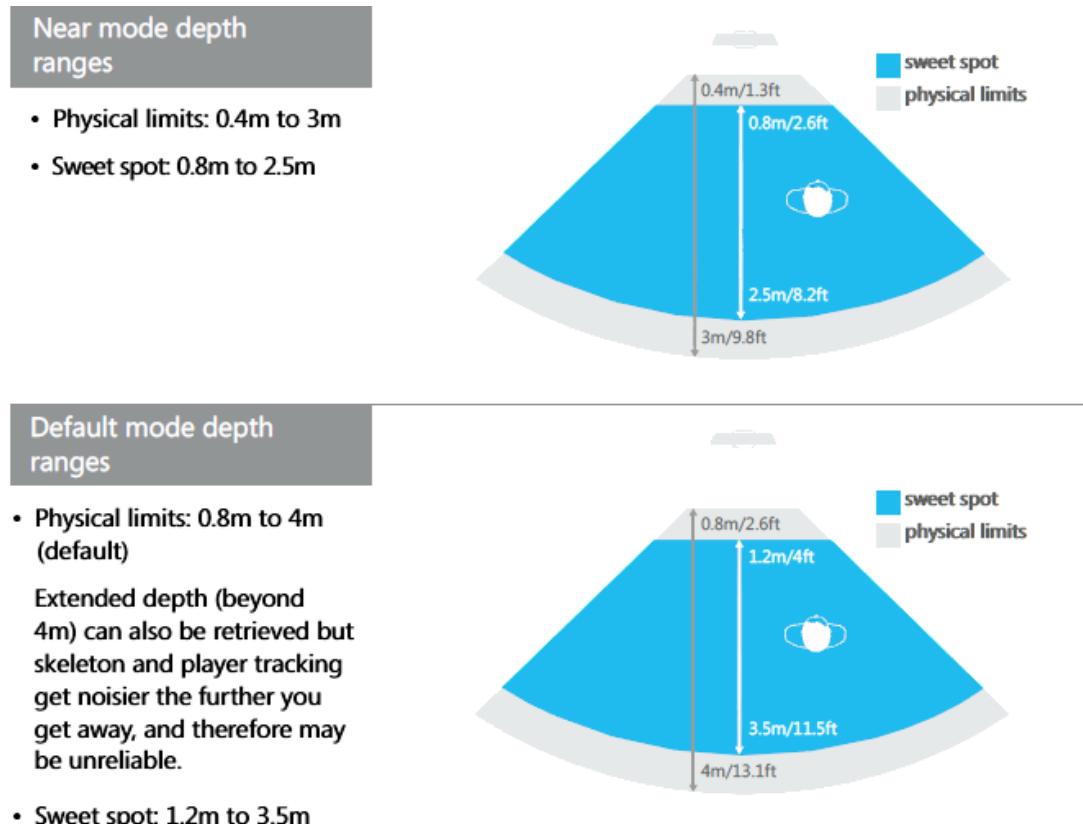


Figura 3.3: Especificación sensores de profundidad.

Aunque con la versión de Kinect Xbox 360 no se puede usar el Near mode depth tampoco es ningún impedimento a la hora del desarrollo del proyecto, ya que nos interesa tener todo el alcance posible y por lo tanto el default mode depth (que está

en todas las versiones) es más apropiado. Una de las mayores limitaciones a la hora de la realización del proyecto serán estos **3.5 metros de profundidad** para el correcto funcionamiento del seguimiento del cuerpo, algo imprescindible para el buen funcionamiento de la aplicación de entrenamiento.

## 2. Cámara RGB.

La cámara RGB (rojo, verde, azul) de la Kinect. Es del tipo CMOS, trabaja por defecto de **640x480 a 30fps**.

De cara al proyecto esta cámara es más que suficiente para el correcto funcionamiento, aunque si que es cierto que para dar una vista al usuario sobre sus movimientos la resolución puede ser algo baja. Es decir, si tuviera una mayor resolución sería más atractivo visualmente. El frame rate parece más que suficiente para cumplir los objetivos del proyecto.

## 3. Micrófono multi matriz

En la parte delantera del sensor kinect hay una serie de micrófonos que se usan para el reconocimiento de órdenes y charla. No son necesarios de cara al proyecto.

## 4. Inclinación motorizada

Una unidad mecánica en la base del sensor de kinect lo inclina automáticamente hacia arriba o hacia abajo según sea necesario. Esta característica, aunque útil, tampoco es vital para el desarrollo del proyecto.

### Angle of vision (depth and RGB)

- Horizontal: 57.5 degrees
- Vertical: 43.5 degrees, with -27 to +27 degree tilt range up and down

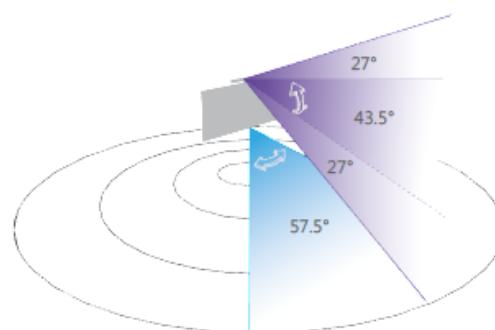


Figura 3.4: Especificación inclinación kinect

## 3.2 Desarrollar para kinect

En este apartado se muestran las alternativas que tenemos para desarrollar para kinect así como la elección final. Explicaremos elementos del SDK más relevantes para el desarrollo del proyecto.

### 3.2.1 Alternativas y elección

- **SDK oficial de Microsoft para Windows [5]**: Apareció en el año 2012, dos años después de la aparición de Kinect después de que Microsoft se diera cuenta del gran interés que generaba. El SDK oficial incluye drivers para usar la kinect en Windows 7 y Windows 8 (aplicaciones de escritorio únicamente), una API, documentación, códigos de ejemplo y diversas herramientas bastante útiles para el desarrollo de kinect como KinectStudio. Para aplicaciones no comerciales es gratuita.
- **LibFreenect (opensource) [6]**: Es un driver creado por la comunidad sin ánimo de lucro Openkinect. Permite obtener los datos de bajo nivel de la kinect, es decir imagen a color y el mapa de profundidad. Apareció en 2010 a través de ingeniería inversa.
- **OpenNi(opensource) [7]**: Driver que apareció en diciembre de 2010 y fue creado por una organización sin ánimo de lucro con el mismo nombre. Además de proporcionar los datos de bajo nivel permite obtener datos de alto nivel, como el seguimiento de cuerpos.

Una vez que hemos presentado las diferentes alternativas vamos a exponer las principales diferencias entre ellas y como afectan al proyecto.

#### Lenguajes de programación soportados:

- SDK oficial de Microsoft: C++, C# y Visual Basic con Microsoft Visual Studio.
- Alternativas opensource: Python, C, C + +, C #, Java, etc.

Vemos que mientras que las alternativas opensource son multiplataforma y multilenguaje el SDK oficial de Microsoft sólo funciona bajo Windows sobre determinados lenguajes.

### **Calibración Kinect:**

- SDK oficial de Microsoft: No necesaria, transparente para el programador.
- Alternativas opensource: Requiere calibrar la Kinect siguiendo las instrucciones que proporcionan.

Un aspecto a tener en cuenta evitar que sea el usuario/programador el que tenga que calibrar de forma correcta la Kinect.

### **Facilidad de uso, documentación y soporte:**

- SDK oficial de Microsoft: Fácil instalación (un click), api sencilla y muy completa. Viene con códigos de ejemplos, está bien documentado y tiene un foro de soporte.
- Alternativas opensource: Instalación más laboriosa. No hay tanta documentación como en el oficial y hay muy pocos ejemplos. OpenNi tiene un foro de soporte, LibFreenect no.

Se nota que Microsoft ha invertido tiempo y dinero en el SDK de Kinect, aunque apareció mucho más tarde tiene una api más completa y un soporte muy por encima de las alternativas libres.

### **Otros aspectos a considerar:**

El driver de Microsoft ofrece mayor precisión y más velocidad a la hora de detectar y hacer el seguimiento de los cuerpos de los usuarios. Además ofrece todas las características que ofrecen las alternativas opensource de forma más sencilla.

Todas las alternativas ofrecen la posibilidad de conectar múltiples Kinects a un mismo PC (el driver de Microsoft a partir del año 2013), aunque esta característica no la vamos a usar en el proyecto.

Finalmente, comentar que el SDK oficial de Microsoft viene con una serie de herramientas muy potentes a la hora de desarrollar para Kinect, como KinectStudio, una herramienta muy útil para el desarrollo del proyecto (y de cualquier otra aplicación con Kinect).

### **Elección para el proyecto:**

A día de hoy vemos que las alternativas opensource son peores en todos los aspectos respecto al SDK oficial, la única característica que aportarían es que son multiplataforma y multilenguaje. Aunque de cara al proyecto estaría bien que fuera multiplataforma, no es un requisito imprescindible y no es motivo suficiente para escoger alguna de las alternativas opensource.

Por si no fuera suficiente la herramienta KinectStudio del SDK oficial es muy útil de cara al proyecto. Permite no tener que hacer la captación de las Katas en tiempo real, pudiendo grabar la realización de una Kata e ir frame a frame a la hora de guardar las posturas que forman una Kata para su posterior visualización/entreno.

Por todo lo comentado en este apartado se ha elegido el **SDK oficial de Microsoft** para el desarrollo del proyecto.

### 3.2.2 SDK oficial de Microsoft

Una vez vistas las diferentes alternativas y decantarnos por el SDK oficial de Microsoft vamos a explicar en este apartado lo que la API nos ofrece, así como sus elementos más relevantes de cara al desarrollo del proyecto.

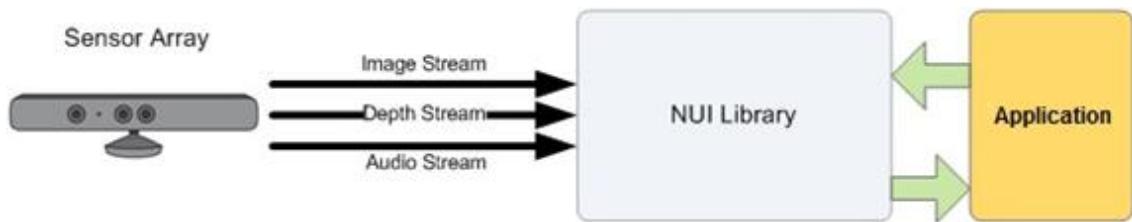


Figura 3.5: Hardware y software interacción con aplicación.

**Nota:** El desarrollo del proyecto se ha hecho con la versión del SDK 1.6.0, durante el que apareció la versión del SDK 1.7.0 con nuevas características no relevantes para el proyecto. Debido a que el proyecto estaba en una fase avanzada decidí no cambiar de versión.

## Arquitectura:

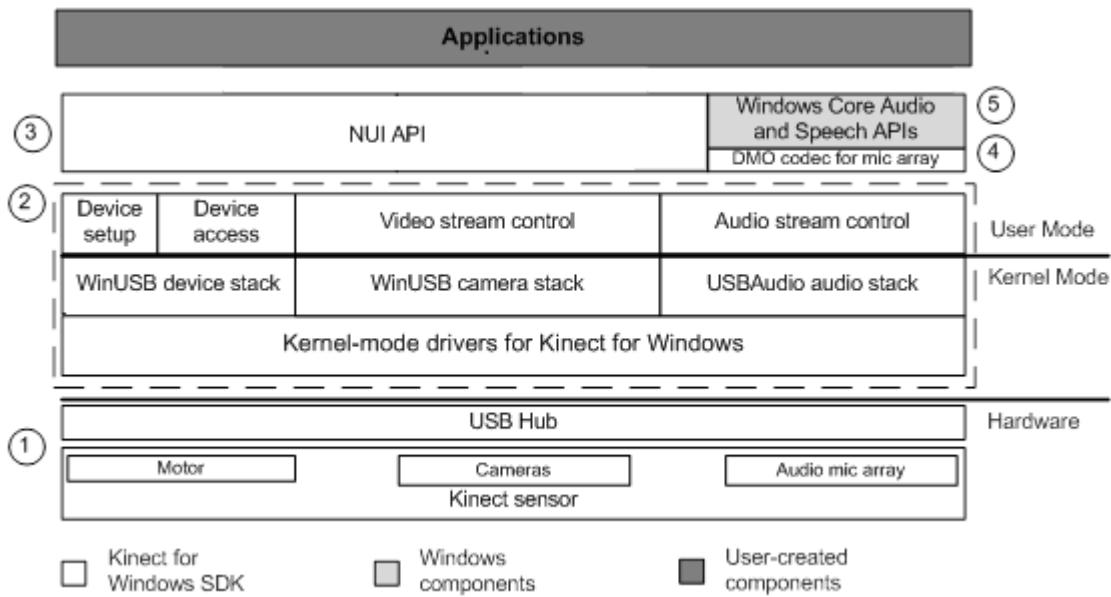


Figura 3.6: Arquitectura.

- 1. Hardware de Kinect** - Los componentes de hardware, incluido el sensor de Kinect y el USB a través del cual el sensor Kinect está conectado al PC.
- 2. Drivers de Kinect** - Los controladores de Windows para la Kinect, que se instala como parte del proceso de instalación SDK.
- 3. NUI API** - conjunto de API que recopila datos de los sensores de imágenes, profundidad y sonido y controla los dispositivos Kinect del sistema.
- 4. DirectX Media Object (DMO)** - matriz de micrófonos (localización de la fuente de audio).
- 5. Windows 7 API estándar** – Api de audio, voz y multimedia de Windows 7 (funciona en Windows 8 modo escritorio).

## Elementos relevantes para el proyecto de la API de Kinect:

**KinectSensor [8]:** Clase que representa un sensor de Kinect (una kinect conectada al ordenador). Es la clase principal de la API de Kinect. Usarla es tan sencillo como comprobar que está en el estado que queremos (conectada y lista para usar), configurar algunos parámetros sobre su funcionamiento y llamar al método `Start()` para que empiece a funcionar en nuestra aplicación. La configuración más importante que tenemos que hacer a esta clase es indicarle a cuantos frames por segundo funcionará y que queremos que nos proporcione cada frame (imagen, profundidad y/o skeletons).

**ColorStream** [9]: Datos que contienen la imagen en color que nos proporciona la Kinect. En cada frame (si lo hemos especificado así en el KinectSensor) podremos acceder a los datos de imagen en color. Esta imagen en color se codifica en RGB, YUV o Bayer según lo configuremos. Esta clase tiene métodos para copiar esta imagen a otras estructuras de datos.



Figura 3.7: Imagen adquirida a través de la clase ColorStream de Kinect.

**DepthStream** [10]: Datos que contienen la imagen de profundidad que capta la Kinect. Cada frame podemos obtener los datos de profundidad que contiene la distancia (en milímetros). Además proporciona una segmentación por usuarios, proporcionándonos un valor entero que indica el índice del usuario de cada píxel de la escena.



Figura 3.8: Imagen obtenida a través de los datos normalizados proporcionados por DepthStream.

**SkeletonStream** [11]: Datos que contienen los “esqueletos” de los usuarios. Cada frame Kinect nos proporciona una array de Skeletons que nos permiten conocer la posición de los usuarios. Kinect permite reconocer hasta 6 usuarios activos y hacer el seguimiento (proporcionar el Skeleton entero) de dos usuarios. De cara al proyecto esto no supone ningún problema, ya que el proyecto esta pensado para un único usuario.

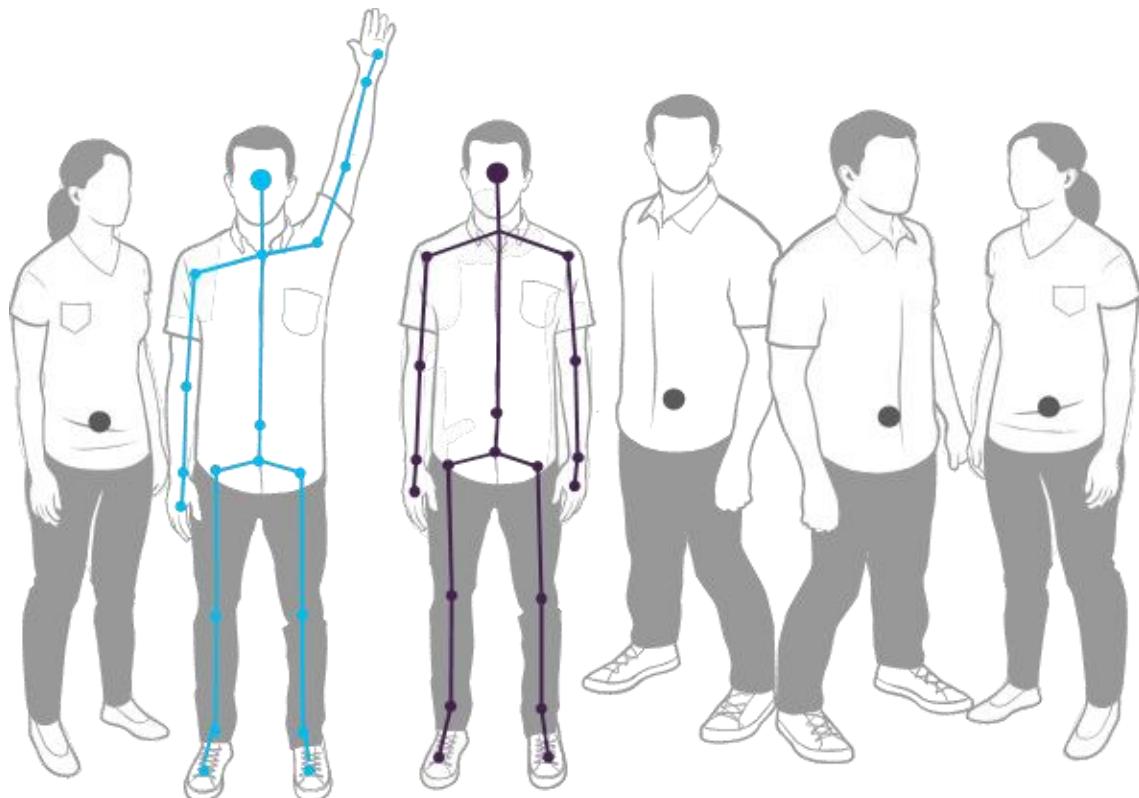
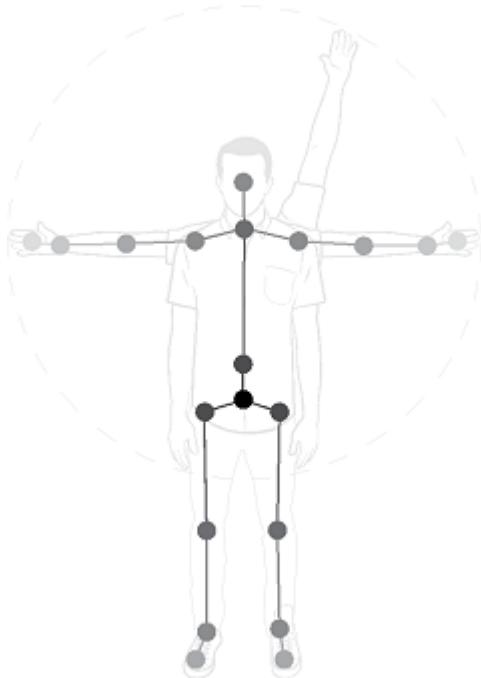


Figura 3.9: Kinect reconoce hasta 6 usuarios y puede hacer el seguimiento (Skeleton entero) de 2.

**Skeleton** [12]: Representa el “esqueleto” de un usuario que capta la Kinect. El Skeleton esta formado por **Joints** [13], que representan diferentes articulaciones del cuerpo. Los Joints nos permiten acceder a las coordenadas (relativas a la kinect) de cada parte del cuerpo del usuario.



Hip Center			
Spine	Hip Left		Hip Right
Shoulder Center	Knee Left		Knee Right
Shoulder Left	Head	Shoulder Right	Ankle Left
Elbow Left		Elbow Right	Ankle Right
Wrist Left		Wrist Right	Foot Left
Hand Left		Hand Right	Foot Right

Figura 3.10: Skeleton compuesto por los Joints que nos proporciona kinect.

La Api del SDK oficial también nos ofrece los métodos para transformar las coordenadas de los joints a coordenadas de las imágenes obtenidas (2D).

Una cosa a tener en cuenta sobre los **Joints** es que tienen diferentes estados, **Tracked** e **Inferred**. El estado tracked significa que la Kinect ha reconocido perfectamente el Joint correspondiente (aunque en algunas ocasiones no sea correcto), sin embargo en el estado Inferred significa que Kinect no ha identificado donde está el joint claramente, y hace una estimación del lugar donde estará que puede ser totalmente incorrecta. De cara al proyecto esto es algo que tenemos que tener muy en cuenta, tanto a la hora de la captación de la kata (evitar posturas con Joints inferred) como a la hora del algoritmo de reconocimiento de posturas.



Figura 3.11: Skeleton con Joints tracked.



Figura 3.12: Skeleton con Joints inferred (pintados de amarillo).

Aunque la API que nos proporciona el SDK oficial de Microsoft tiene muchas más clases importantes (por ejemplo, el control de audio) aquí solo se han explicado por encima aquellas relevantes para el desarrollo del proyecto.

### 3.2.3 KinectStudio

KinectStudio [14] es una herramienta que permite grabar y reproducir secuencias de profundidad y color captadas con Kinect. Las reproducciones se pueden hacer en cualquier aplicación que use el SDK oficial de Kinect, por lo que podemos reproducir cualquier secuencia grabada con KinectStudio en cualquier aplicación.

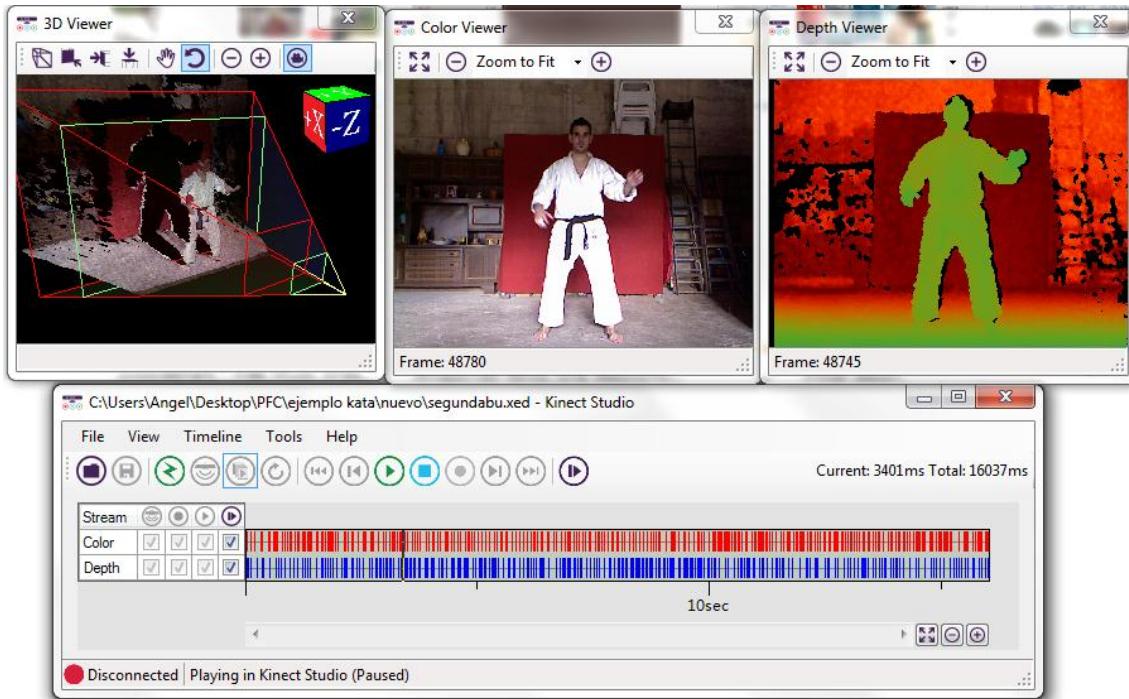


Figura 3.13: KinectStudio mostrando una secuencia grabada.

Esta herramienta es uno de los principales motivos por lo que se ha escogido el SDK oficial de Microsoft por encima de las otras alternativas ya que es realmente potente. En cualquier aplicación ayuda mucho a la hora de depurar el programa y permite crear escenarios repetibles para su testeo, pero es que además es una herramienta idónea para el proyecto ya que permite reproducir una secuencia frame por frame.

De cara a nuestro proyecto nos permite captar una Kata de forma controlada, no en tiempo real mientras el experto realiza la kata. Dicho de otra manera, podemos grabar una kata de un experto y reproducirla de forma pausada, frame por frame, para captar con total precisión y sin errores las posturas que queremos que formen la kata.

### 3.3 Sistema operativo

En este punto ya hemos decidido el uso del SDK oficial de Microsoft para la Kinect, por lo que no tenemos alternativa. Las aplicaciones del proyecto se desarrollarán para **Windows 7**, aunque es compatible con Vista y Windows 8 (modo escritorio).

### 3.4 Lenguaje de programación

Al igual que la elección de sistema operativo, la elección del lenguaje de programación viene condicionada por el uso del SDK oficial de Microsoft para la Kinect. El SDK oficial únicamente soporta C++, C# y Visual Basic con Microsoft Visual Studio.

Cualquiera de estas alternativas es válida para el desarrollo de las aplicaciones del proyecto, y aunque dispongo de experiencia en C++ me ha parecido una buena oportunidad aprovechar el proyecto para familiarizarme con un nuevo lenguaje para mí como es **C#**. De esta forma mientras se desarrolla el proyecto puedo aprender un nuevo lenguaje.

### 3.5 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) [15] permite el desarrollo de interfaces en Windows con los lenguajes de programación .NET. Ofrece una amplia infraestructura y potencia gráfica con la que es posible desarrollar aplicaciones visualmente atractivas. Permite integrar elementos de animación, vídeo, imágenes y audio de manera sencilla, cosa que viene bien para el desarrollo de este proyecto.

WPF utiliza el lenguaje XAML para proporcionar un modelo declarativo para la interfaz de interacción. XAML es un lenguaje declarativo basado en XML, algo parecido a los layouts de android.

Una de las partes más potentes de WPF es el Databinding, que permite separar la interfaz de interacción de la lógica del negocio. Esto permite por ejemplo que cada vez que haya un cambio en el modelo de nuestra aplicación se refleje en la vista de forma transparente para el programador.

Dado la elección del lenguaje de programación, el IDE a usar y sobre el sistema operativo en el que se va a desarrollar la aplicación me ha parecido la elección más lógica.

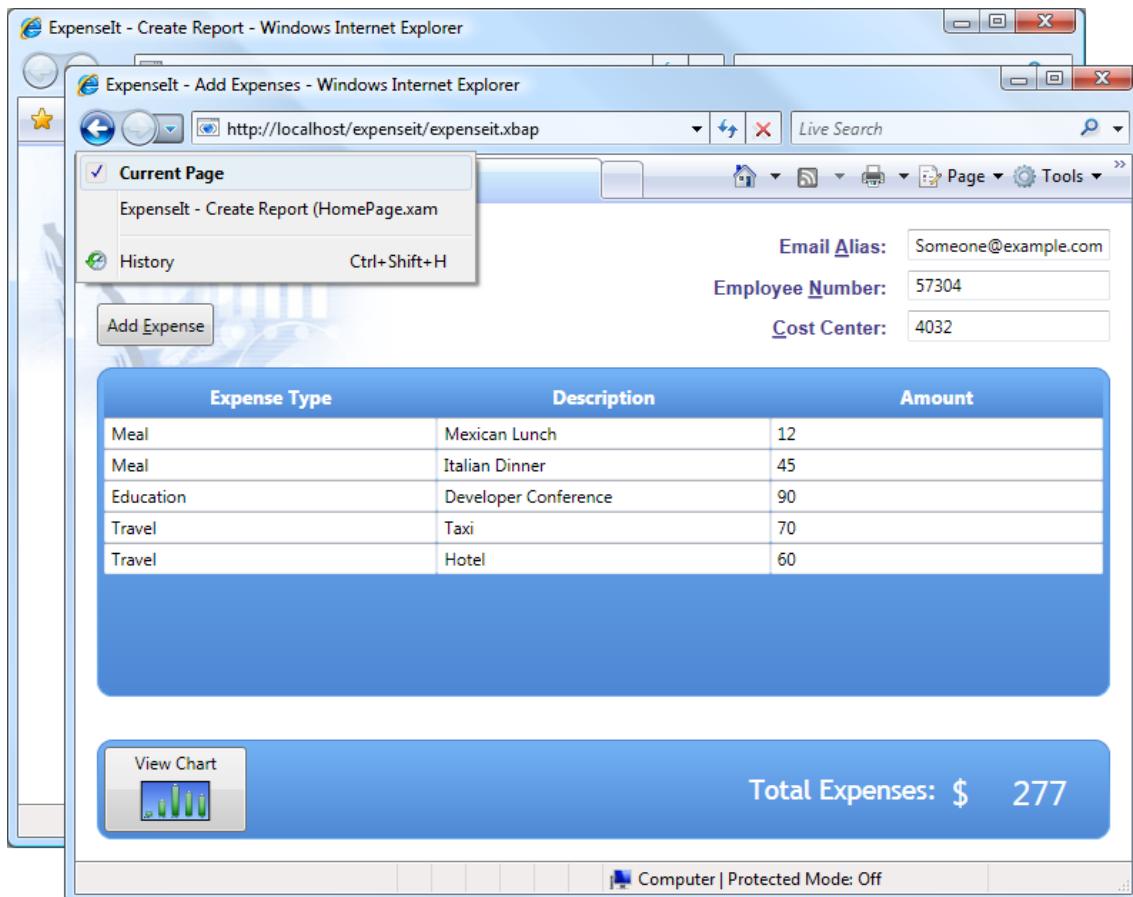


Figura 3.14: Ejemplo interfaz desarrollada con WPF.

### 3.5.1 Otra alternativa

La otra alternativa que se ha barajado para el desarrollo de la aplicación ha sido usar la interfaz que ofrece XNA [16]. XNA es un entorno de programación que permite usar Visual Studio 2010 para crear juegos para ordenadores con Windows, para Xbox360 y para Windows phone.

La idea fue finalmente descartada porque no aportaba ningún valor al desarrollo del proyecto y añadía dificultad innecesaria en algunos aspectos.

# 4. Especificación

En este apartado vamos a definir el alcance del proyecto, así como extraer, definir y detallar las características más importantes que se han de considerar para el desarrollo de las aplicaciones.

Antes de empezar a definir características y delimitar el alcance se revisaran algunos conceptos que considero imprescindibles para el desarrollo del proyecto.

## 4.1 Conceptos relevantes para el desarrollo del proyecto

### Kata (karate):

Kata significa “forma”. A nivel básico, es una sucesión de técnicas de bloqueo y golpeo determinadas (fijas) que se ejecutan al aire contra enemigos imaginarios. Todo el volumen de técnicas, tácticas y algunos apartados de acondicionamiento físico para la práctica del karate se encuentran resumidos en los katas.

Para el ámbito de este proyecto, una kata para nosotros será una **sucesión de posturas de karate determinada**.

### Postura (karate):

Para el ámbito de este proyecto definiremos postura como una **posición relevante de una técnica de bloqueo o golpeo de karate**. Parte del peso del proyecto consiste en la captación de estas posturas a través de la Kinect para formar una kata y el reconocimiento de estas posturas para poder entrenar estas katas.

### Skeleton:

Un *skeleton* es un conjunto de *Joints* (coordenadas respecto a la kinect) que nos indica el lugar de las articulaciones de una persona que nos proporciona la API del SDK oficial de Microsoft para la kinect. Esto nos permitirá definir posturas.

## 4.2 Definición de alcance

El alcance de un proyecto consiste en contestar que se va incluir y que no se va a incluir en el proyecto, marcando una frontera clara. Esto permite delimitar las funcionalidades de la aplicación, dando una idea exacta al equipo de trabajo que se tiene que realizar y evitar esfuerzos y tiempo en aspectos que no entran dentro del alcance.

El alcance de este proyecto está condicionado por el plazo de entrega de PFC, por lo que definiremos las funcionalidades imprescindibles que tienen que tener las aplicaciones.

Sin entrar al detalle, el alcance incluye:

- Desarrollo de *KataCreator*, aplicación que permita captar una kata tal y como la hemos definido en los conceptos del proyecto a través de los datos obtenidos de la Kinect.
- Guardar en disco la kata obtenida con *KataCreator*.
- Desarrollo de *KataTraining*, aplicación que permita cargar, visualizar y entrenar las katas obtenidas con *KataCreator*.
- Reconocimiento de posturas (tal y como la hemos definido en los conceptos del proyecto)

El alcance no incluye:

- Uso de gráficos de ordenador para representar posturas de karate. Todos los elementos multimedia serán obtenidos de imágenes reales.
- Reconocimiento de factores diferentes a las posturas que pueden ser importantes a la hora de realizar una kata (como puede ser la respiración), ya que Kinect no nos permite controlarlo.

## 4.3 Características de *KataCreator*

Esta aplicación está pensada para la captación de las katas que posteriormente se usarán para visualizarse o entrenarse en la aplicación de *KataTraining*. Aunque esto podría estar incluido en una única aplicación junto con *KataTraining* se ha decidido que sean aplicaciones separadas porque no se quiere permitir que cualquier usuario añada Katas para entrenar en *KataTraining*, sino que sean personas con cierto nivel en karate las que creen las katas para que tengan un nivel de calidad satisfactorio.

La aplicación consistirá en una única vista que tiene que:

- Informar de si la Kinect está conectada y lista para usar.
- Mostar en todo momento la imagen que capta la kinect, con el *skeleton* del usuario que aparezca por ella.
- Permitir añadir la postura actual (mostrada en el punto anterior) que realiza el usuario.
- Mostrar, en otra imagen, la postura actual de la Kata con su *skeleton*.
- Permitir eliminar la postura actual de la kata.
- Permitir navegar entre las posturas de la kata.
- Rellenar información sobre la kata, nombre, dificultad, cinturón, etc.
- Guardar la Kata en disco.
- Minimizar/cerrar aplicación.

## 4.4 Características de KataTraining

Esta aplicación está pensada para la visualización o entrenamiento de las Katas grabadas con *KataCreator*. Esta aplicación tiene que ser sencilla e intuitiva y permitir que el usuario seleccione la kata para poder visualizarla o entrenarla.

La aplicación consistirá en varias vistas, la vista principal tiene que:

- Cargar y mostrar la lista de Katas creadas con *KataCreator*, mostrando su nombre, dificultad, etc.
- Permitir ordenar por dificultad las Katas.
- Visualizar una kata, que abrirá una vista su visualización.
- Entrenar una kata, que abrirá una vista su entrenamiento.
- Minimizar/cerrar aplicación.

La vista de visualización tiene que:

- Mostrar, en orden, las posturas que forman la kata que se ha seleccionado para visualizar.
- Mostrar el nombre de la Kata.
- Minimizar/cerrar ventana.

La vista de entrenamiento tiene que:

- Informar al usuario de si la Kinect está conectada.
- Indicar cuando puede el usuario empezar a realizar la kata.
- Mostrar el nombre de la kata.
- Mostar en todo momento la imagen que capta la kinect, con el *skeleton* del usuario que aparezca por ella.
- Mostrar al usuario las siguientes dos posturas de la Kata a realizar por parte del usuario.
- Reconocer, en tiempo real, si la postura del usuario captada por la kinect se corresponde con la postura de la kata a realizar, y en caso afirmativo actualizar las posturas a realizar por las siguientes.
- Minimizar/cerrar ventana.

## 4.5 Actores

Los actores para la aplicación *KataCreator* son la Kinect con e sistema, el usuario que interactúa con la aplicación de escritorio y el usuario que está enfrente de la Kinect. Este último puede ser sustituido por de una grabación de KinectStudio.

El actor para la aplicación *KataTraining* es el usuario que escogerá una kata para visualizarla o entrenarla. En el entrenamiento de la Kata la interacción con la aplicación se hace a través de la Kinect.

## 4.6 Decisiones preliminares

Una vez tenemos claro lo que queremos hacer y que elementos tenemos para llevar el proyecto a cabo vamos a terminar de definir algunos aspectos sobre el desarrollo del proyecto.

La **persistencia** de los datos se hará en ficheros **XML** en disco, teniendo en cuenta que para guardar algunas imágenes lo haremos en ficheros .png. Dado que el proyecto únicamente guardará katas e información de la kata que no será editable desde la aplicación de entrenamiento (*kataTraining*) me ha parecido fuera de lugar usar un sistema de persistencia más complejo como podría ser el uso de base de datos.

# 5. Desarrollo

En el apartado anterior se han especificado las funcionalidades y el comportamiento de las aplicaciones que se quieren crear. Esto se ha hecho de forma textual explicando lo que se quería conseguir sin entrar en detalle. En este apartado se explicará el desarrollo de las aplicaciones, definiendo los detalles de cómo se llevan a cabo las funcionalidades y el comportamiento de las aplicaciones.

## 5.1 Metodología

Para llevar a cabo el desarrollo del proyecto se ha dividido el trabajo en bloques, de manera que a la hora de implementar me he podido centrar en un conjunto de funcionalidades reducidas de las aplicaciones (teniendo en cuenta como se tienen que integrar con el resto). Una de las partes más importantes del proyecto es el reconocimiento de posturas a través de la Kinect por lo que se ha decidido tratarlo como uno de los primeros bloques (tal y como podemos ver en la planificación).

Para el desarrollo de las funcionalidades más críticas del proyecto nos hemos basado en la metodología Test Driven Development [17] (TDD), primero escribir el test unitario y después implementar la funcionalidad. Seguir esta metodología me ha permitido la tarea del desarrollo por bloques asegurándonos que el resto del código funciona correctamente.

## 5.2 Borradores de vistas

Antes de empezar a detallar los casos de uso se mostrarán unos borradores de las vistas de la aplicación que nos ayudará a visualizar lo que se quiere realizar. Explicaremos los elementos importantes que contienen las vistas.

### 5.2.1 Vista de la aplicación KataCreator

Como ya hemos comentado anteriormente, esta aplicación consta de una única vista que nos tiene que permitir captar una secuencia de posturas que forman una kata para poder guardarla y posteriormente usarla en la aplicación *KataTraining*.

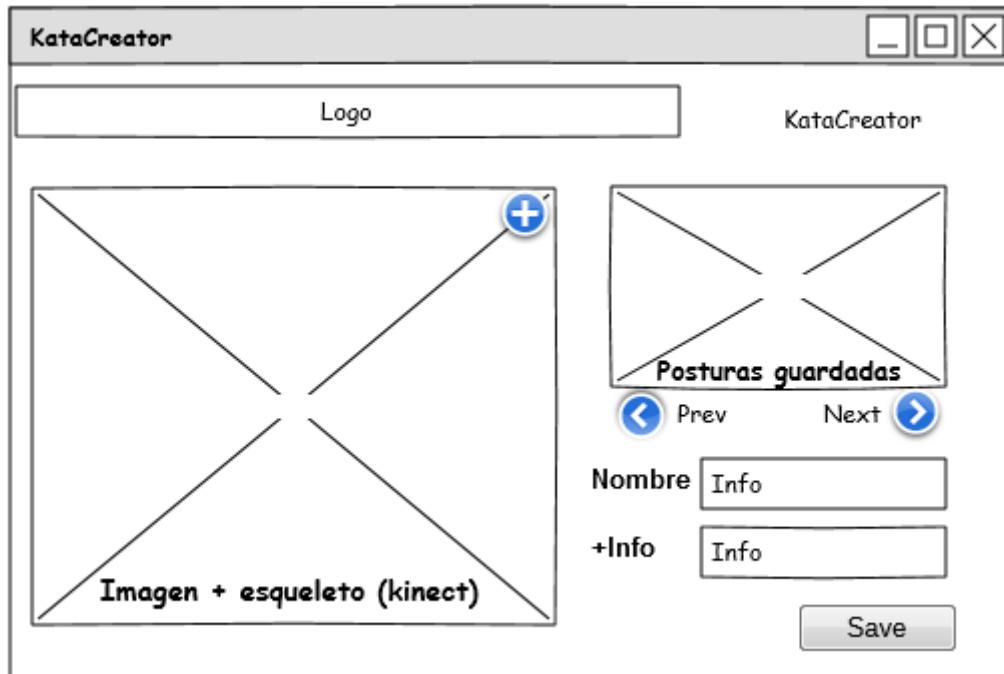


Figura 5.1: Vista principal de KataCreator.

Los elementos más importantes de esta vista son:

- Nombre de la aplicación y logo.
- Imagen que muestra lo que capta la Kinect. En caso de reconocer a un usuario nos mostrará también su *skeleton*. Es la imagen más grande con el nombre de “Imagen + *skeleton* (Kinect)” que se sitúa en la izquierda en la figura 5.1.
- Botón (representado con un + en la figura 5.1) para añadir la postura actual que este captando la Kinect en la kata.
- Imagen que muestra la última postura añadida en la Kata. Es la imagen más pequeña con el nombre de “posturas guardadas” que se encuentra en la derecha de la figura 5.1.
- Botón para eliminar la postura actual que se muestra de la Kata.
- Botones para avanzar y retroceder por las posturas de la Kata representados por < > en la figura 5.1.
- Cajas de texto para llenar información de la kata: nombre, descripción ...
- Botón para guardar la Kata actual.
- Botones para cerrar y minimizar la aplicación.

### 5.2.2 Vistas de KataTraining

Al contrario que *KataCreator* esta aplicación tiene más de una vista. Se quiere que esta aplicación sea fácil de usar para cualquier usuario independientemente de sus conocimientos tecnológicos, por lo que estas vistas tienen que ser muy sencillas e intuitivas.

La aplicación consta de tres vistas, la vista principal para seleccionar la kata que queremos visualizar o bien entrenar, la vista de visualización para que el usuario de la aplicación pueda visionar la kata escogida y la vista de entrenamiento en la que el usuario puede entrenar la kata seleccionada.

A continuación se muestra el borrador de la vista principal de *KataTraining*.

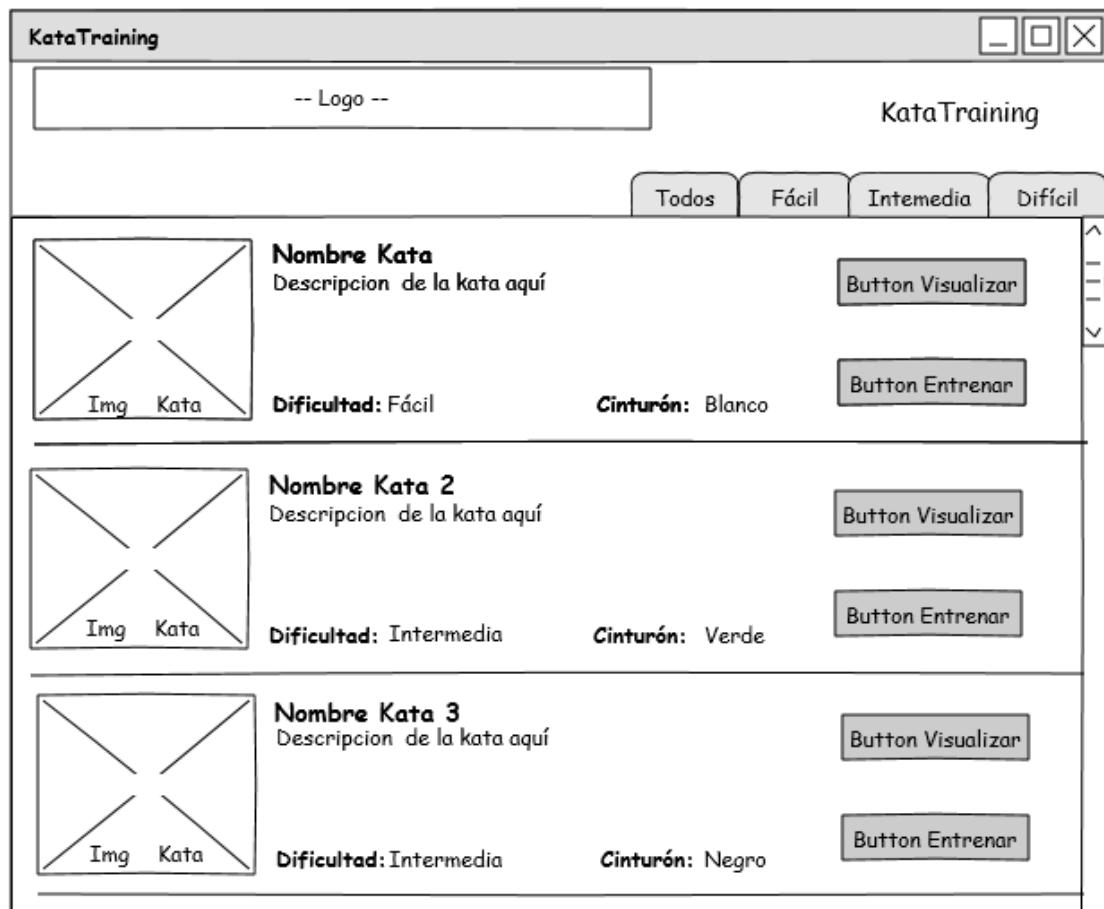


Figura 5.2: Vista principal de KataTraining.

Los elementos más importantes de esta vista son:

- Nombre de la aplicación y logo.
- Lista de botones para seleccionar las katas por dificultad: Todos, Fácil, Intermedia y difícil.
- Lista de Katas para visualizar/entrenar.
- Scroll para desplazarse por la lista de katas en caso necesario.
- Botones para cerrar y minimizar la aplicación.

Las katas que se muestran en esta vista tienen los siguientes elementos:

- Imagen representativa de la kata (la imagen con el nombre de “Img Kata” en la figura 5.2).
- Información sobre la kata: Nombre de la Kata, una descripción, dificultad y cinturón.
- Botones de Visualizar/Entrenar por cada kata.

A continuación se muestra el borrador de la vista de entrenamiento.

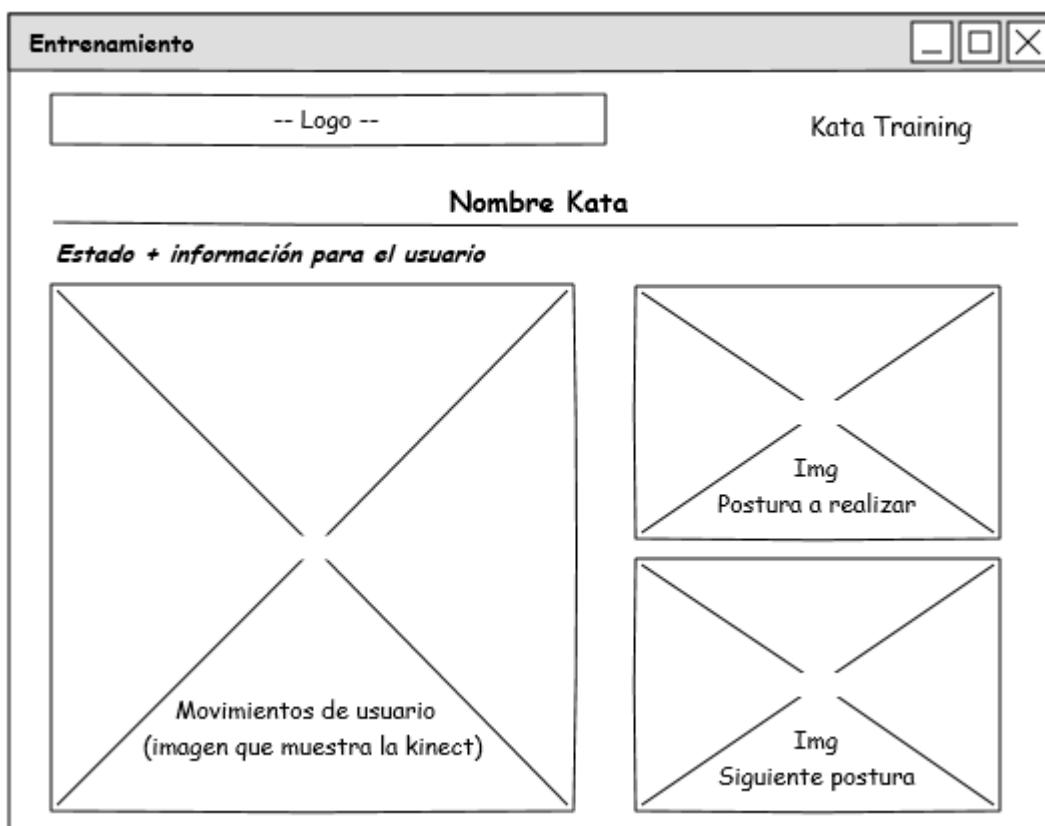


Figura 5.3: Vista de entrenar de KataTraining.

Los elementos más importantes de esta vista son:

- Nombre de la vista, logos y nombre de la aplicación.
- Nombre de la kata.
- Texto que indica el estado de la Kinect o información sobre la manera correcta de realizar la kata para el usuario.
- Imagen (izquierda en la figura 5.3) que muestra lo que capta la kinect, en caso de reconocer algún usuario muestra su *skeleton*.
- Dos imágenes ubicadas a la derecha en la figura 5.3 que contienen las siguientes dos posturas a realizar de la kata para que el usuario las pueda observar. Estas imágenes cambiaran conforme el usuario vaya realizando de las katas que muestran.
- Botones para cerrar y minimizar la ventana.

Finalmente se muestra el borrador de la vista de visualización.

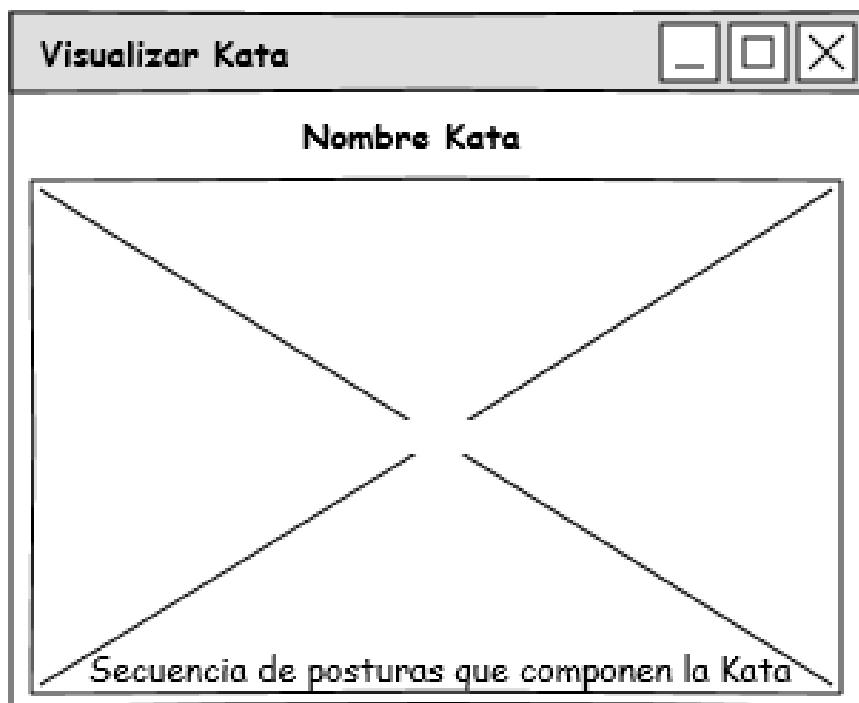


Figura 5.4: Vista de visualizar de KataTraining.

Los elementos más importantes de esta vista son:

- Nombre de la vista.
- Nombre de la kata.
- Imagen que muestra las diferentes posturas que forman la kata.
- Botones para cerrar y minimizar la ventana.

## 5.3 Casos de uso

En esta sección se explican las interacciones entre los actores y el sistema para nuestras aplicaciones. Aunque para ser del todo precisos habría que considerar también la interacción de la Kinect con el sistema aquí por simplicidad únicamente se muestran los casos de uso en los que el usuario interacciona de forma directa sobre el sistema.

Ya que el proyecto consta de dos aplicaciones veremos sus casos de uso por separado.

### 5.3.1 Diagrama interacción Usuario Sistema de KataCreator:

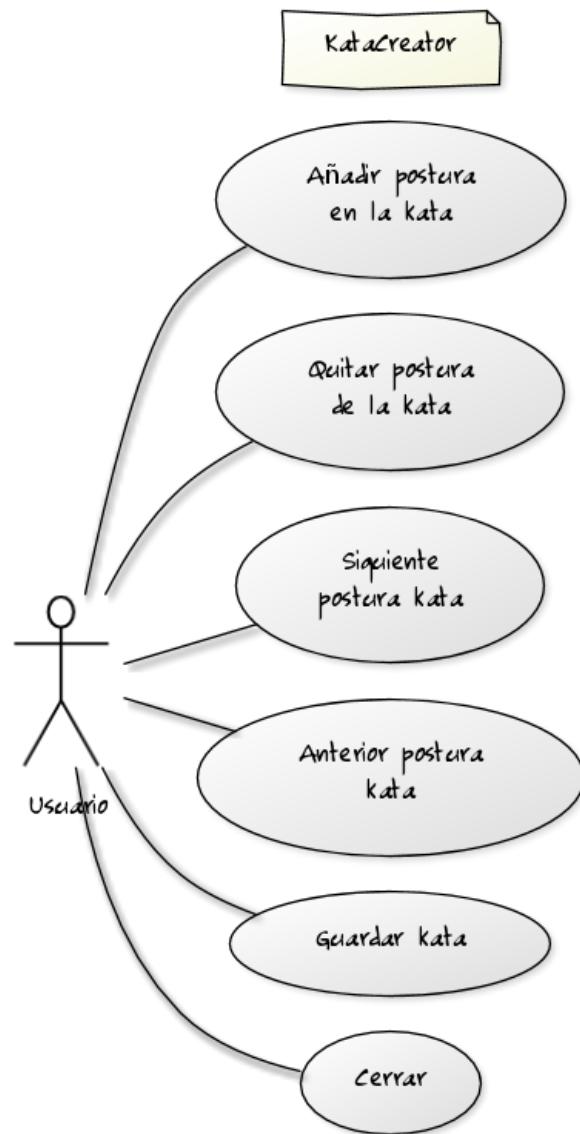


Figura 5.5: Diagrama interacción usuario sistema de KataCreator.

### **Añadir postura en la kata:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere añadir la postura actual que recoge la Kinect en la kata. El sistema añade la nueva postura en la kata y actualiza la vista mostrando al usuario que la nueva postura ha sido añadida correctamente.

**Caso de uso:** Añadir postura en la kata

**Actores:** Usuario

#### **Flujo básico:**

- El usuario indica al sistema que quiere añadir la postura que capta la Kinect en la kata.
- El sistema añade la postura actual que ofrece la Kinect como la siguiente postura en la kata.
- El sistema actualiza la imagen que muestra la nueva postura añadida como la postura actual de la kata.

#### **Flujo alternativo 1:**

- El usuario indica al sistema que quiere añadir la postura que capta la Kinect en la kata.
- El sistema no hace nada si la Kinect no está conectada.

#### **Flujo alternativo 2:**

- El usuario indica al sistema que quiere añadir la postura que capta la Kinect en la kata.
- El sistema informa mediante un pop-up que la Kinect aún no ha reconocido a un usuario activo del que captar la postura.

### **Quitar postura de la kata:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere quitar la postura actual de la kata. El sistema quita la postura de la kata y la siguiente postura (si existe) pasa a ser la postura actual.

**Caso de uso:** Quitar postura de la kata

**Actores:** Usuario

**Flujo básico:**

- El usuario indica al sistema que quiere quitar la postura actual que forma la kata.
- El sistema elimina postura actual de la kata.
- El sistema actualiza la postura actual de la kata por la siguiente postura, en caso de que exista, y actualiza la vista para reflejárselo al usuario.

**Flujo alternativo 1:**

- El usuario indica al sistema que quiere quitar la postura actual que forma la kata.
- En el sistema no hay ninguna postura añadida en la kata, por lo que no hace nada.

**Siguiente postura kata:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere avanzar la postura actual de la kata a la siguiente. El sistema actualiza la vista mostrando la siguiente postura si existe. Si se encuentra en la última postura de la kata pasa a mostrar la primera.

**Caso de uso:** Siguiente postura kata

**Actores:** Usuario

**Flujo básico:**

- El usuario indica al sistema que quiere avanzar la postura actual de la kata a la siguiente postura.
- El sistema actualiza la imagen que permite visualizar las posturas de la kata y muestra la siguiente imagen que forma la kata, o muestra la primera si la postura actual es la última postura de la kata.

**Flujo alternativo 1:**

- El usuario indica al sistema que quiere avanzar la postura actual de la kata a la siguiente postura.
- Si la kata no tiene posturas el sistema no hace nada.

### **Anterior postura kata:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere retroceder la postura actual de la kata a la anterior. El sistema actualiza la vista mostrando la anterior postura si existe. Si la postura actual es la primera postura de la kata pasa a mostrar la última.

**Caso de uso:** Anterior postura kata

**Actores:** Usuario

#### **Flujo básico:**

- El usuario indica al sistema que quiere retroceder la postura actual de la kata a la anterior postura.
- El sistema actualiza la imagen que permite visualizar las posturas de la kata y muestra la anterior imagen que forma la kata, o muestra la última si la postura actual es la primera postura de la kata.

#### **Flujo alternativo 1:**

- El usuario indica al sistema que quiere retroceder la postura actual de la kata a la anterior postura.
- Si la kata no tiene posturas el sistema no hace nada.

### **Guardar kata:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere guardar la kata en disco. El sistema guarda en disco la kata actual con las posturas que el usuario ha añadido.

**Caso de uso:** Guardar Kata

**Actores:** Usuario

**Flujo básico:**

- El usuario indica al sistema que quiere guardar la kata en disco.
- El sistema abre una ventana estándar para saber en qué ruta se guardará la kata.
- El sistema guarda en disco la kata actual así como la información que hemos rellenado sobre la kata en la ruta seleccionada.

**Flujo alternativo 1:**

- El usuario indica al sistema que quiere guardar la kata en disco.
- El sistema informa a través de un pop-up que no hay ninguna postura en la kata e indica al usuario como añadir posturas.

**Flujo alternativo 2:**

- El usuario indica al sistema que quiere guardar la kata en disco.
- El sistema informa a través de un pop-up que no se ha rellenado toda la información disponible de la kata y pide al usuario que los rellene antes de guardarla en disco.

**Cerrar:**

Esta interacción se efectúa en la única vista de *KataCreator* cuando el usuario indica al sistema que quiere cerrar la aplicación. La aplicación se cierra.

**Caso de uso:** Cerrar

**Actores:** Usuario

**Flujo básico:**

- El usuario indica al sistema que quiere cerrar la aplicación.
- El sistema libera todos los recursos (la Kinect entre otras cosas).
- La aplicación se cierra.

### 5.3.2 Diagrama interacción Usuario Sistema de KataTraining:

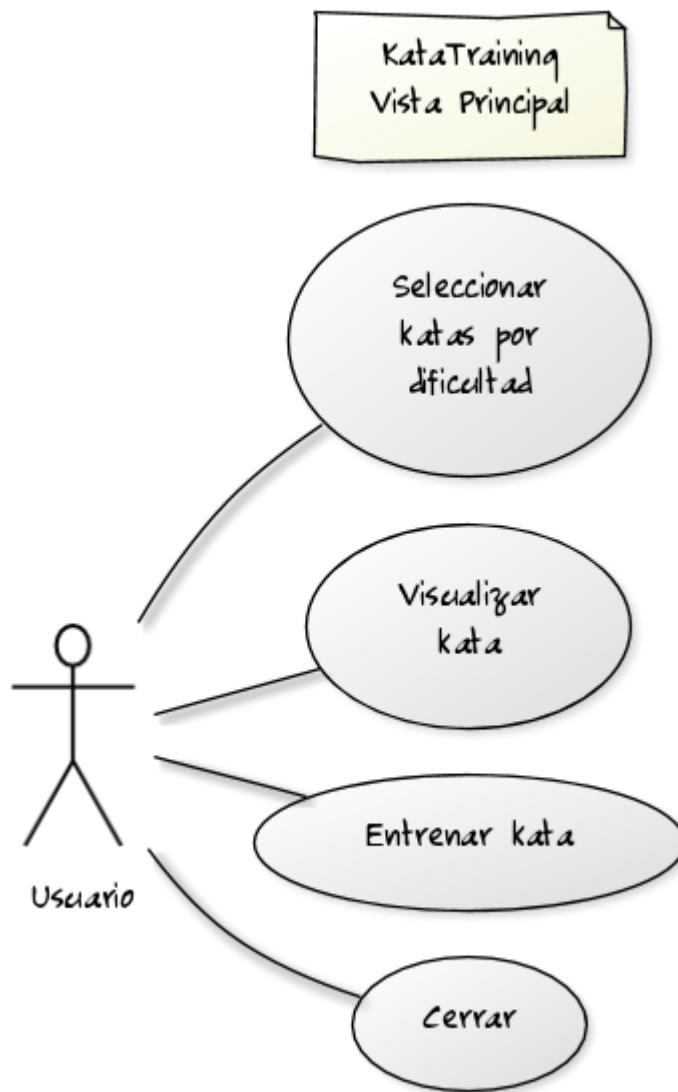


Figura 5.6: Diagrama interacción usuario sistema de KataTraining.

#### Seleccionar katas por dificultad:

Esta interacción se efectúa en la vista principal de *KataTraining* cuando el usuario indica al sistema que quiere seleccionar las katas de determinado nivel de dificultad. El sistema pasa a cargar y mostrar las katas que se corresponde con el nivel seleccionado.

**Caso de uso:** Seleccionar katas por dificultad

**Actores:** Usuario

**Flujo básico:**

- El sistema carga la información de todas las katas disponibles y se las muestra al usuario.
- El usuario indica al sistema que quiere filtrar las katas por dificultad.
- El sistema carga únicamente la información de las katas que tengan el nivel de dificultad seleccionado y muestra la lista al usuario.

**Flujo alternativo 1:**

- El sistema falla al cargar la información de las katas, muestra al usuario una lista de katas vacía.

**Visualizar kata:**

Esta interacción se efectúa en la vista principal de *KataTraining* cuando el usuario indica al sistema que quiere visualizar una kata de la lista de katas que el sistema muestra. El sistema abre una nueva ventana donde el usuario puede visualizar las posturas que forman una kata.

**Caso de uso:** Visualizar kata.

**Actores:** Usuario

**Flujo básico:**

- El sistema carga la información de todas las katas disponibles y se las muestra al usuario.
- El usuario indica al sistema que quiere visualizar una de estas katas.
- El sistema abre una nueva ventana donde el usuario puede visualizar en orden las posturas que forman una kata.

**Flujo alternativo 1:**

- El sistema carga la información de todas las katas disponibles y se las muestra al usuario.
- El usuario indica al sistema que quiere visualizar una de estas katas.
- El sistema abre una nueva ventana pero falla a la hora de cargar la kata que queremos visualizar.
- El usuario es informado y puede cerrar esta ventana.

### **Entrenar kata:**

Esta interacción se efectúa en la vista principal de *KataTraining* cuando el usuario indica al sistema que quiere entrenar una kata de la lista de katas que el sistema muestra. El sistema abre una nueva ventana donde el usuario puede entrenar una kata.

**Caso de uso:** Entrenar kata.

**Actores:** Usuario

### **Flujo básico:**

- El sistema carga la información de todas las katas disponibles y se las muestra al usuario.
- El usuario indica al sistema que quiere entrenar una de estas katas.
- El sistema abre una nueva ventana donde el usuario puede entrenar, a través de la Kinect, una kata.

### **Flujo alternativo 1:**

- El sistema carga la información de todas las katas disponibles y se las muestra al usuario.
- El usuario indica al sistema que quiere entrenar una de estas katas.
- El sistema abre una nueva ventana pero falla a la hora de cargar la kata que queremos entrenar.
- El usuario es informado y puede cerrar esta ventana.

En la vista de entrenar una kata el usuario sólo puede interactuar con el sistema a través de la Kinect, y aunque esto no deja de ser otro caso de uso hemos comentado que únicamente se explican los casos de uso en el que el usuario tiene interacción directa con el sistema, y no a través de la Kinect. No obstante creo importante comentarlo para tenerlo presente.

**Cerrar:**

Esta interacción se efectúa en la vista de *KataTraining* cuando el usuario indica al sistema que quiere cerrar la aplicación. La aplicación se cierra.

**Caso de uso:** Cerrar**Actores:** Usuario**Flujo básico:**

- El usuario indica al sistema que quiere cerrar la aplicación.
- El sistema libera todos los recursos.
- La aplicación se cierra.

## 5.4 Descripción de clases

En esta sección se describen las clases más relevantes de las dos aplicaciones. No se pretende entrar al detalle explicando todos sus atributos ni sus métodos, no obstante en este documento sí que se pretende dar una idea general, de forma textual, de cada clase. Comentar que algunas clases coinciden para las dos aplicaciones, únicamente las veremos una vez.

### 5.4.1 Clases del modelo que comparten las dos aplicaciones

**Kata:**

Una de las clases más importante del modelo. Representa una kata de karate que contiene una lista ordenada de las posturas que la forman. Contiene métodos para agregar una postura a la kata, quitar una postura de la kata, dar acceso a la postura actual de la kata y para navegar por sus posturas entre otras cosas.

**ItemKata:**

Clase que contiene información que describe una Kata. Nombre, descripción, dificultad, etc. Esta información está separada de la clase Kata porque al mostrar en *KataTraining* la lista de katas disponibles para visualizar o entrenar únicamente queremos cargar esta información, no el contenido de las katas.

#### **Postura:**

Clase que representa una posición relevante de una técnica de golpeo o defensa de karate. Los atributos que la forman son la imagen de la postura captada, el *Skeleton* que proporciona Kinect (para poder mostrarlo en la vista) y el cálculo de información necesaria para el algoritmo de reconocimiento de posturas extraída de los datos proporcionados por la Kinect.

#### **SkeletonInfo:**

Clase que contiene el cálculo de información que usa el algoritmo de reconocimiento de posturas, más concretamente una lista de distancias normalizadas de cada “articulación” (*Joint*) que proporciona la Kinect respecto a la cintura (el centro de seguimiento de Kinect).

#### **Dist:**

Clase que contiene la distancia normalizada de un *Joint* del usuario que proporciona la Kinect a su cintura. Contiene además otra información relevante como el tipo de *Joint* al que pertenece y si es o no inferred (es inferred si la kinect supone su posición, no la sabe con exactitud en algunos casos).

### 5.4.2 Clases de KataCreator

#### **MainWindow:**

Clase que implementa la interfaz gráfica de la vista principal de *KataCreator*. Carga los elementos definidos en MainWindow.xaml (recordemos que estamos usando WPF) y captura la pulsación de los diferentes botones. Si la Kinect está conectada esta clase es la responsable de mostrar cada frame la imagen que capta la Kinect y dibujar su *Skeleton*.

#### **MainWindowViewModel:**

Controlador Vista-Modelo de la aplicación que consta de una única vista. Esta clase es la que interactúa entre MainWindow y el modelo.

#### **GestorDisco:**

Clase singleton que se encarga de dar persistencia en disco de la clase Kata y de *ItemKata*. Lo hace en ficheros .xml. La aplicación *KataCreator* no tiene que cargar ningún elemento de disco.

### 5.4.3 Clases de KataTraining

#### **MainWindow:**

Clase que implementa la interfaz gráfica de la vista principal de *KataTraining*. Es la encargada de mostrar al usuario la lista de katas (con su Nombre, descripción, cinturón, etc) para que pueda escoger la que quiere entrenar o visualizar.

#### **ListaKatas:**

Clase que hereda de ObservableCollection, una clase que permite hacer binding entre los datos del modelo y la vista para la lista de ItemKata que muestra la información disponible de las katas. Esta clase es necesaria porque para hacer binding de una lista en WPF es la única herramienta de la que se dispone. Esto permite que cuando los datos del modelo se modifican la vista se actualice de forma transparente para el programador.

#### **PrevisualizarWindow:**

Clase que implementa la interfaz gráfica de la vista de visualización de la Kata. Esta vista muestra una imagen por donde se ven las posturas que forman una kata. Esta vista no permite ninguna interacción por parte del usuario.

#### **PrevisualizarWindowViewModel:**

Controlador Vista-Modelo de la vista de visualización. Esta clase es la que permite cargar y acceder la Kata seleccionada para poder visualizarla desde la vista.

#### **EntrenamientoWindow:**

Clase que implementa la interfaz gráfica de la vista de entrenamiento de la Kata. Carga todos los elementos necesarios para poder interactuar a través de la Kinect.

### **EntrenamientoWindowViewModel:**

Controlador Vista-Modelo de la vista de entrenamiento. Esta clase es la que permite cargar y acceder a la Kata para poder entrenarla.

### **CheckStance:**

Una de las clases más importantes del proyecto, contiene el **algoritmo para comparar dos posturas** y decidir si corresponden o no a la misma postura. A esta clase se le llama cada frame en el que la kinect obtiene un nuevo *Skeleton* para compararlo con la postura actual de la Kata.

Creo importante explicar por encima el algoritmo ya que es una de las piezas claves del proyecto. En un primer momento, y después de informarme sobre el tema, se barajó la posibilidad de usar técnicas de *machine learning* que parece ser que funcionan bastante bien. Esta opción se rechazo por falta de muestras para el aprendizaje, las personas que hacen karate y me han ayudado en el desarrollo del proyecto tendrían que haber pasado una gran cantidad de tiempo delante de la Kinect para diferentes katas. También dificultaba mucho a la hora de crear nuevas katas ya que si se introducían nuevos movimientos habría que volver a pasar por otra fase de aprendizaje.

Finalmente se ha optado por un algoritmo que se basa en la distancia normalizada de los diferentes *Joints* que proporciona la Kinect respecto al punto central de seguimiento de los usuarios, la cintura.

Este algoritmo tiene una serie de ventajas e inconvenientes que vamos a explicar. Entre sus ventajas vemos que no es muy costoso computacionalmente, si tenemos en cuenta que la comparación de posturas la hacemos sobre 30 veces por segundo esto es un elemento a tener en cuenta. Otra ventaja es que al tener la distancia normalizada no nos afectan elementos como la distancia respecto a la Kinect (siempre y cuando se capte el cuerpo entero) ni la altura de los usuarios. Entre sus desventajas nos encontramos con que la orientación de la postura si que afecta a la hora de hacer la comparación. Otra de sus desventajas es tener que decidir un “margen de error” que vamos a permitir para considerar dos posturas iguales (demasiado grande puede considerar dos posturas diferentes como la misma, demasiado pequeño puede hacer que dos posturas muy similares sean consideradas diferentes).

Existen otros algoritmos, la mayoría basados también en distancias normalizadas de los *Joints* (como por ejemplo comparar la distancia normalizada de todos los *Joints* con todos). Al ser más costosos computacionalmente se han descartado.

### GestorDisco:

Clase singleton que se encarga de cargar de disco la clase Kata e ItemaKata. La aplicación *KataTraining* no tiene que guardar ni modificar ningún elemento del disco.

## 5.5 Pruebas y resultados

Una vez que tenemos el sistema implementado hemos realizado una serie de pruebas para ver y valorar su funcionamiento. Dado que este proyecto se basa en la construcción de un sistema que interacciona entre un usuario y la Kinect se ha considerado que la mejor manera de evaluar el resultado es crear diferentes katas con usuarios reales y probar que funcionen de forma satisfactoria.

### 5.5.1 Participantes

Para realizar las pruebas he contado con la ayuda de tres personas, todas ellas practican Karate en diferentes categorías. Una de ellas es cinturón negro (el cinturón más alto, segundo dan), otra es cinturón azul y la última persona lleva muy poco tiempo practicando karate y es amarillo (uno de los más bajos).



Figura 5.7: Participantes que hacen karate.

### 5.5.2 Descripción general de las pruebas

La forma de probar la aplicación consiste en grabar, mediante el uso de KinectStudio, muchas repeticiones de algunas katas. De esta forma guardaremos la misma kata varias veces. Una vez tengamos muchas repeticiones de una misma kata por parte de los tres usuarios crearemos nuestra kata con *KataCreator* del usuario que mayor nivel tiene. Finalmente nos queda probar con *KataTraining* la Kata creada anteriormente con las repeticiones restantes, empezando por las repeticiones hechas por el mismo usuario del que se ha captado la Kata y siguiendo con las repeticiones del resto de usuarios. Esto lo haremos para diferentes katas.

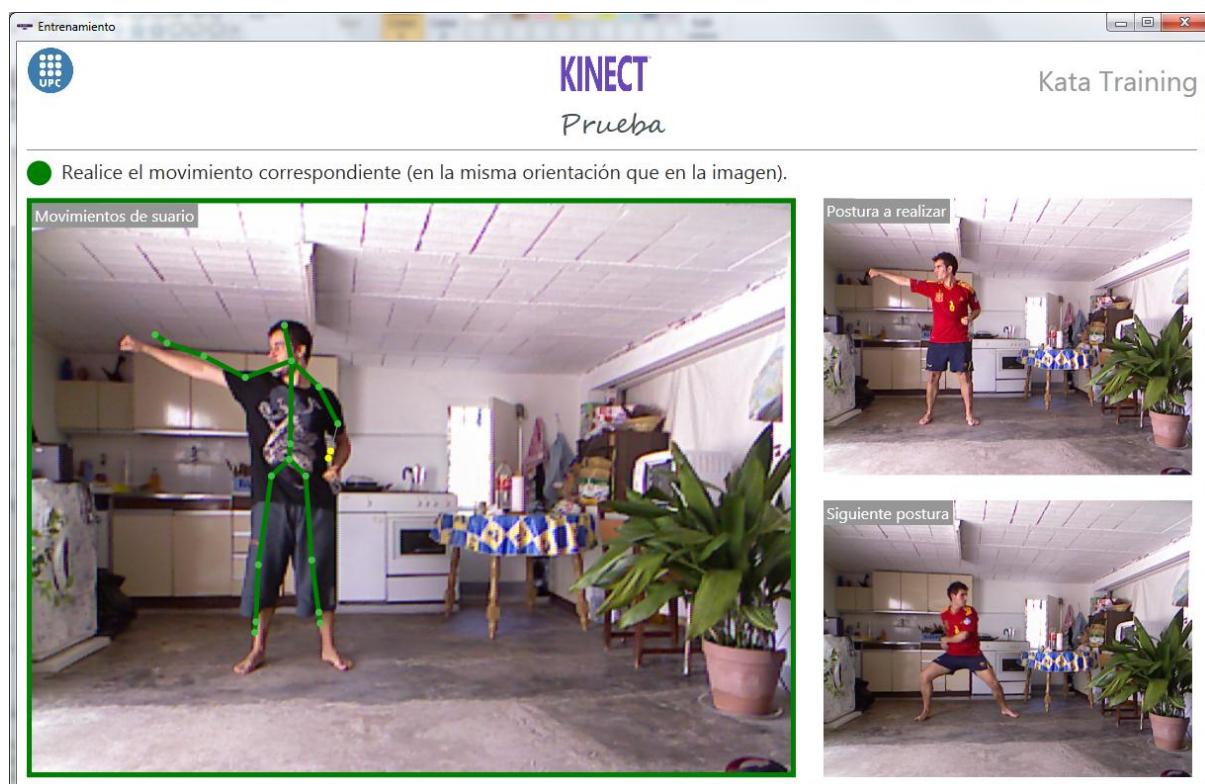


Figura 5.8: Participante realizando kata (captada con KataCreator) en KataTraining.

### 5.5.3 Consideraciones y restricciones previas

A causa de algunas limitaciones de la Kinect y de las aplicaciones desarrolladas en el proyecto se han puesto algunas restricciones a la hora de realizar las pruebas. A continuación se describen las más importantes:

- Muchas katas necesitan mucho espacio para su realización, el usuario se tiene que mover mucho en diferentes direcciones. Debido a la limitación de la Kinect que llega hasta los 3,5-4 metro (y para captar el cuerpo entero de una persona tiene que estar sobre los 3 metros aproximadamente) la selección de las katas a probar queda bastante limitada.



Figura 5.9: Usuario realizando una kata que se sale fuera del rango de la Kinect.

- Todas las katas se han realizado en interiores. El SDK oficial de Microsoft para Kinect funciona realmente mal en exteriores con luz natural por lo que se ha evitado.
- Por simplicidad se han grabado todas las katas en entornos similares, misma distancia.

- Algunas posturas de las katas pueden dar problemas, por ejemplo una patada a la altura del hombro puede ser mal reconocida por el SDK de Kinect (al ser una postura poco natural). Se han evitado estas y otras posturas en los que el SDK de Kinect puede tener problemas (evitando, por ejemplo posturas de espaldas a la Kinect).



Figura 5.10: Postura de usuario mal reconocida por Kinect.

#### 5.5.4 Resultados de las pruebas

Desde el punto de vista del funcionamiento correcto del reconocimiento de las posturas de las katas y después de probar muchas repeticiones de la lista de katas sobre las que hemos realizado las pruebas, se observa que podemos crear Katas para su posterior entreno de forma satisfactoria para katas que cumplen las restricciones comentadas en el apartado anterior, donde las posturas son claramente identificables por el SDK de Microsoft para Kinect. Esto implica que las posturas tienen que ser claras (una postura no puede tapar parte del cuerpo del usuario) y hay que tener especial cuidado con algunas posturas poco naturales (como una patada por encima del hombro). Esto último lo podemos ver a la hora de captar la kata gracias a la representación visual del *Skeleton*.

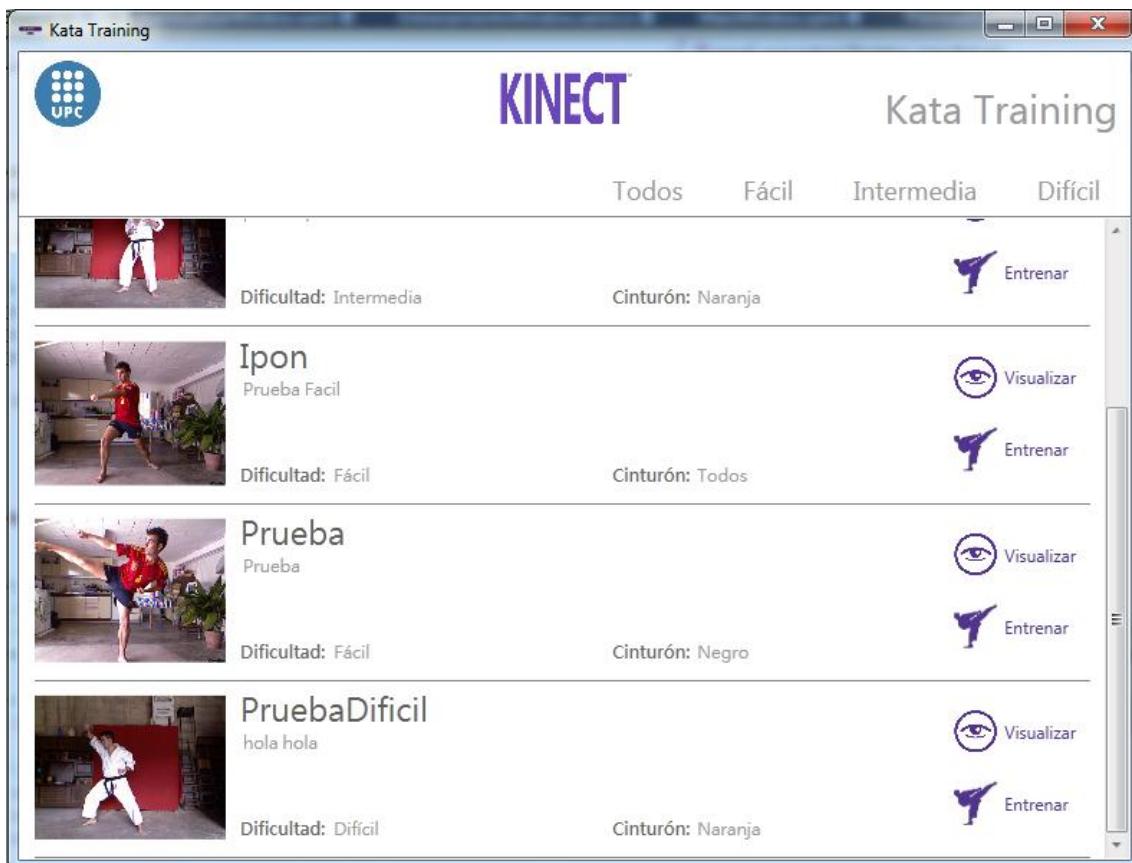
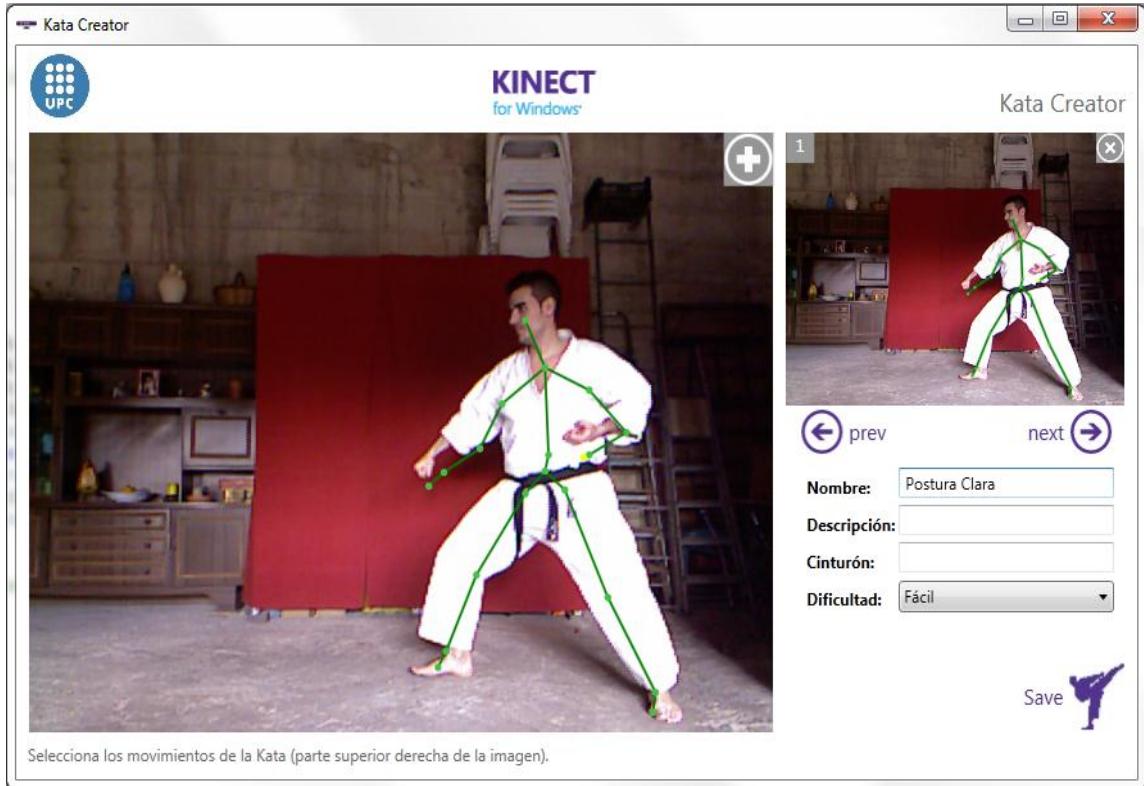


Figura 5.11: Vista principal de KataTraining con algunas de las katas de las que se han hecho pruebas.

Las katas que no cumplen las restricciones previas mencionadas en el apartado anterior suelen fallar a la hora de detectar una determinada postura haciendo la experiencia poco satisfactoria y útil de cara a practicar una determinada kata, obligando a los usuarios a tener que repetir algunas posturas de las katas en algunos casos o incluso imposibilitando su realización.



**Figura 5.12: Creando kata con KataCreator, postura fácilmente reconocible por la Kinect.**

Desde el punto de vista de la facilidad de uso de las aplicaciones por parte de los usuarios no ha surgido ningún problema, todos las han usado de manera intuitiva. Por un lado tenemos *KataCreator* (captura de Katas), fácil e intuitivo de utilizar. Informa de forma textual o visual de los pasos para crear la kata e indica si no se cumplen la mayoría de restricciones necesarias para el correcto funcionamiento comentadas en el apartado anterior. Por el otro lado, la aplicación *KataTraining* es aún más fácil e intuitiva de usar, basta con escoger la kata y colocarse enfrente de la Kinect para realizarla.

En resumen, los participantes han podido usar de forma intuitiva las aplicaciones y crear diferentes katas (teniendo en cuenta las restricciones) que han podido entrenar con un reconocimiento de las posturas de las katas funcional.

# 6. Planificación y presupuesto

Este apartado muestra la planificación que se ha seguido durante la planificación del proyecto y un análisis económico del coste total que habría supuesto la realización del proyecto fuera del ámbito de un proyecto final de carrera.

## 6.1 Planificación

Aunque el proyecto se inscribió a finales de junio de 2012 no fue hasta octubre de 2012 cuando se empezó su desarrollo. Además, durante el desarrollo del proyecto, he estado acabando algunas asignaturas finales de carrera y trabajando en una beca de colaboración con el departamento de Matemática aplicada II. Por estos motivos la realización del proyecto ha sido lenta en algunos meses y no se matriculo hasta febrero de 2013.

Antes de mostrar el diagrama de Gantt en el que se puede observar la planificación se explica de forma breve la planificación por mes:

- **Octubre 2012:** Se empieza a investigar sobre Kinect. Se decide el driver de Kinect que mejor se adapta al proyecto, el lenguaje de programación, que se usará para la interfaz gráfica y el SO sobre el que se desarrollará el proyecto. Durante este mes se empieza el aprendizaje del entorno de trabajo con lo decidido, el SDK de Microsoft para trabajar con Kinect y otras herramientas como KinectStudio.
- **Noviembre 2012:** Al inicio de este mes se establece la idea general del proyecto y los principales objetivos. Se hace un pequeño análisis y diseño de la arquitectura general del sistema. Se comienza a desarrollar una primera versión de KataCreator para captar posturas que forman una kata.
- **Diciembre 2012:** Se acaba la primera versión de KataCreator para captar posturas y se empieza a desarrollar una de las piezas clave del proyecto, el algoritmo para comparar posturas. Se aprovecha que ese cuatrimestre estaba cursando Visió per Computador (VC) en la FIB para su desarrollo.
- **Enero 2013:** Se termina de ajustar el algoritmo de reconocimiento de posturas y se graban, usando KinectStudio, muchas repeticiones de katas de Karate de diferentes usuarios para usarlas durante el desarrollo del proyecto.

- **Febrero 2013:** Se matricula el PFC. Se retoca KataCreator para ajustar nuevas funcionalidades que han surgido, se empieza a desarrollar la vista principal de la aplicación para el entrenamiento de las katas (KataTraining) y se elabora el informe previo.
- **Marzo 2013:** Se continua con el desarrollo de la aplicación de entrenamiento de katas (Kata training), se hace la vista de visualización de las katas y se empieza con la memoria del proyecto.
- **Abril 2013:** Se hace la vista de entrenamiento de una kata de KataTraining y se acaba el desarrollo de la aplicación. Se continúa con la memoria del proyecto. Se comienzan a realizar diferentes pruebas con usuarios reales
- **Mayo 2013:** Se finalizan las pruebas con usuarios y la memoria del proyecto. Se prepara la presentación del proyecto.

En vista de estar trabajando en la beca de colaboración ya dejé un margen generoso para los diferentes bloques en la planificación inicial por lo que apenas se ha visto afectada, aunque han surgido una serie de imprevistos (como por ejemplo la falta de documentación a la hora de hacer binding de una imagen en WPF). La única cosa que se ha visto muy afectada han sido las pruebas finales con usuarios que se han adelantado porque en la planificación inicial estaban demasiado atrasadas. Los objetivos opcionales no se han podido llevar a cabo debido a la limitación de la fecha de entrega del proyecto.

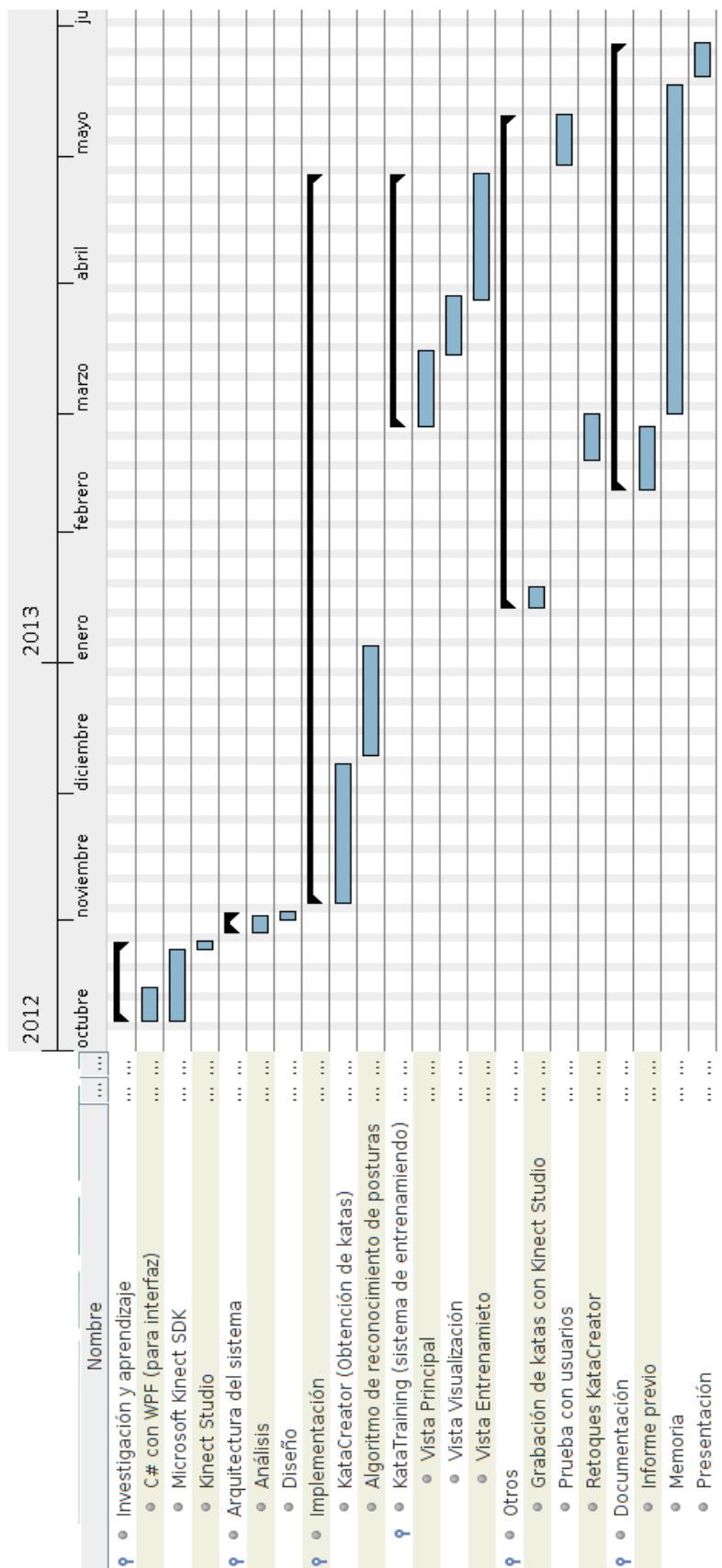


Figura 6.1: Diagrama de Gantt de la planificación del proyecto.

## 6.2 Presupuesto

Este apartado muestra el presupuesto del proyecto suponiendo que no se tratará de un proyecto final de carrera. El presupuesto está dividido en hardware, software y finalmente en coste de personal.

### 6.2.1 Coste de hardware

El equipamiento hardware necesario para llevar a cabo el proyecto han sido un ordenador de gama media y una Kinect para Windows (en caso de aplicación comercial). En la siguiente tabla se muestra el coste.

	Amortización	Coste	Subtotal
Ordenador gama media	25%	600€	150€
Kinect para Windows	100%	250€	250€
<b>Total:</b>			400€

### 6.2.2 Coste de software

Aquí se muestra el software de pago necesario para el desarrollo del proyecto. Por un lado tenemos la licencia del SO Microsoft Windows 7 y la licencia comercial del SDK para kinect.

	Coste
Microsoft Windows 7	90€
Licencia Comercial Kinect	Incluida con Kinect para PC
<b>Total</b>	90€

### 6.2.3 Coste del personal

Aquí se muestra el coste de los empleados del proyecto, con el coste por hora de cada uno de los roles.

	Horas	Coste/hora	Total
Analista	64	50€	3.200€
Programador	584	30€	17.520 €
<b>Total:</b>			20.720€

### 6.2.4 Coste total

Si sumamos todos los costes vemos que nos sale un total de 21.210€. En el ámbito del proyecto final de carrera nos hemos ahorrado todos los costes de licencias para uso comercial y se disponía de todo el material necesario para el desarrollo del proyecto.

Concepto	Coste
Hardware	400€
Software	90€
Personal	20.720€
<b>Total</b>	<b>21.210€</b>

# 7. Conclusiones

En este apartado se revisan los objetivos del proyecto y que es lo que finalmente se ha llevado a cabo. Se comentan también limitaciones debido al hardware de la Kinect que nos hemos encontrado y finalmente se explican ideas de como continuar el proyecto que no han dado tiempo a realizar durante el desarrollo del mismo debido a la limitación de tiempo.

## 7.1 Revisión de objetivos y conclusiones generales

Uno de los objetivos del proyecto a nivel personal ha sido trabajar con tecnologías nuevas para mí que no había tocado con anterioridad. Este aspecto se puede considerar satisfecho del todo, a lo largo del proyecto se ha trabajado con C# (con Windows Presentation Foundation para la interfaz de usuario), el SDK oficial de Microsoft para la Kinect y diferentes herramientas que se han explicado en este documento.

El objetivo principal a nivel de aplicación, crear un sistema de entrenamiento con Kinect para katas de Karate, se puede considerar cubierto siempre y cuando sean katas simples que cumplan los requisitos comentados en el apartado de resultados. Se han desarrollado dos aplicaciones, una para captar las katas (*KataCreator*) y la otra para poder visualizarlas y entrenarlas (*KataTraining*). Sin embargo, durante el desarrollo del proyecto han aparecido una serie de limitaciones de la Kinect que han impuesto unas restricciones en el proyecto. Para este proyecto la mayor limitación de la Kinect es la distancia hasta la que funciona, para captar un cuerpo entero (algo importante en Karate) el usuario se tiene que situar sobre los 3 metros de distancia respecto a la Kinect y la Kinect funciona hasta los 4 metros (aunque en las especificaciones recomiendan hasta los 3,5 metros). Existen otras limitaciones que para nuestro proyecto no han sido relevantes pero que para crear sistemas de entrenamiento con Kinect para otros deportes (como por ejemplo atletismo) pueden ser muy importantes, por ejemplo el mal funcionamiento de la Kinect en exteriores con luz natural.

Para ver sistemas de entrenamiento parecidos al desarrollado en este proyecto 100% funcionales (sin las limitaciones de la Kinect) aún habrá que esperar unos pocos años, por lo menos sin usar sistemas de captación de movimientos mucho más complejos y caros (que ya existen en la actualidad). Ya está anunciada la Kinect 2, que comentan que tendrá una precisión muy superior a la actual con una mayor área de visión.

## 7.2 Trabajo futuro

Vamos a explicar algunas ideas y propuestas interesantes que han surgido antes o durante el desarrollo del proyecto:

- **Uso de lentes sobre Kinect** para hacer más grande el área de entrenamiento. Parece ser que existen unas lentes que se pueden poner en las ópticas de la Kinect que permiten hacer zoom a cambio de deformar la imagen. Como ya hemos comentado una de las mayores limitaciones de la Kinect para el entrenamiento de karate es la distancia hasta la que al Kinect capta bien. Estaría bien probar si estas lentes realmente permiten ganar algo de área de entrenamiento sin deformar la imagen en exceso.
- **Uso de dos o más Kinects** para la captación y reconocimiento de posturas. A la hora de captar una kata se han de evitar posturas que puedan tapar parcialmente parte del cuerpo del usuario. Esto complica mucho la realización de katas que se desplazan en varias direcciones. Si obtuviéramos los datos de varias Kinects colocadas en distintos sitios se podría minimizar este problema. Además durante el desarrollo del proyecto actualizaron el SDK oficial de Kinect para poder trabajar con distintas Kinects conectadas a un mismo ordenador de forma sencilla.
- **Uso del reconocimiento de voz** [18]. Podría ser interesante añadir el reconocimiento de voz de la Kinect para algunas características no disponibles. Por ejemplo, dar la opción a un usuario que esté practicando una kata de volver a empezar/pausar/reanudar usando la voz.

- **Mejorar información mientras se realiza una Kata.** Tal y como se ha desarrollado el proyecto el usuario que está realizando la Kata sabe únicamente si hace correctamente las posturas que la aplicación va indicando. Sería interesante que considerara otro tipo de información como un indicador del tiempo entre posturas, que mostrará de forma visual que partes de la postura son incorrectas, etc.

# 8. Manual de usuario

## 8.1 Requisitos del sistema

- **Requisito de sistema operativo:**
  - Windows 7
  - Windows 8
  - Windows Embedded Standard 7
  - Windows Embedded Standard 8
- **Requisitos mínimos de hardware:**
  - Procesador de 32-bit (x86) o 64-bit (x64)
  - Dual-core, 3-GHz
  - USB 2.0 bus para conectar Kinect
  - 2 GB de RAM
  - Tarjeta gráfica que soporte DirectX 9.0c
  - Un sensor Kinect para Windows
- **Requisitos de software:**
  - Driver oficial de Microsoft para Kinect.
  - .Net Framework 4.0/4.5

## 8.2 Manual de KataCreator

### 8.2.1 Instalación

Antes de poder empezar a usar *KataCreator* se requiere pasar por un sencillo proceso de instalación. Vamos a detallar el proceso paso por paso.

1. Ir a la carpeta donde se encuentra el instalador de *KataCreator* y ejecutar el Instalador KataCreator.msi

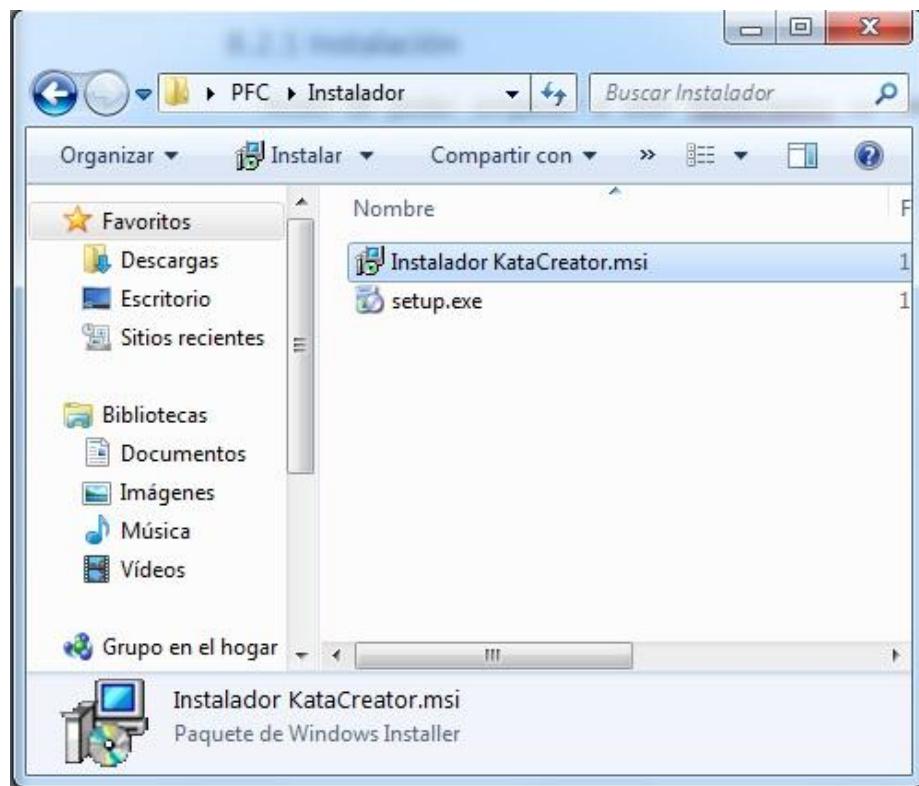


Figura 8.1: Ejecutar Insatalador KataCreator .msi

2. Se abrirá el asistente de instalación, seleccione siguiente.

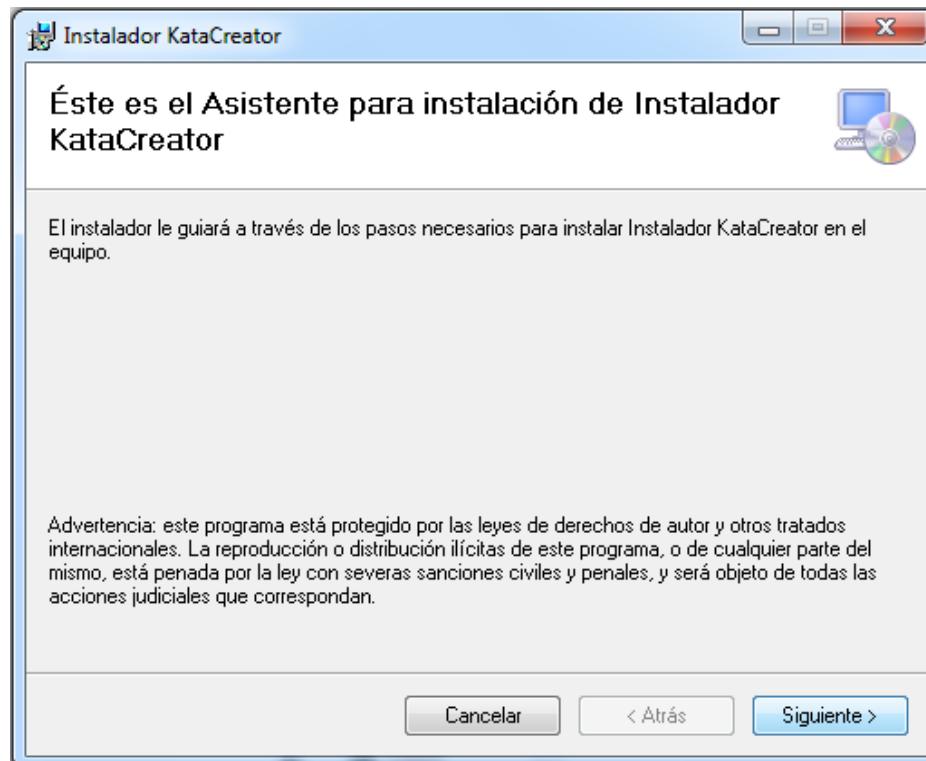


Figura 8.2: Asistente de instalación.

3. Elija la ruta donde se instalará la aplicación.

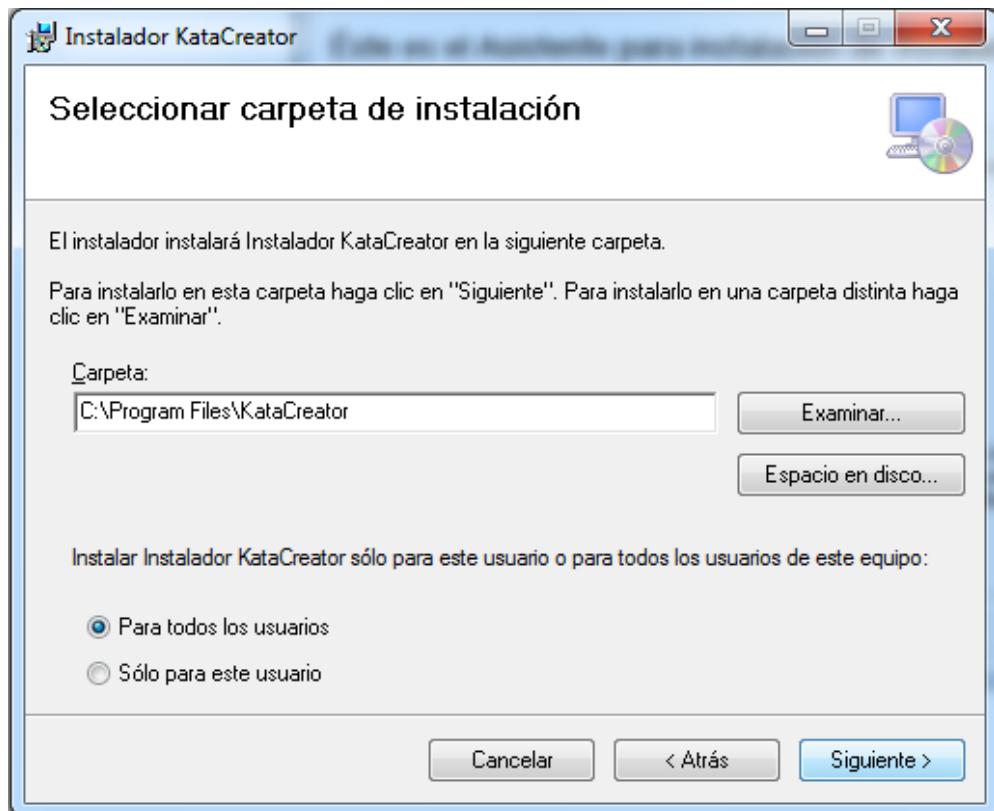


Figura 8.3: Seleccione ruta de instalación.

4. Confirma la instalación.

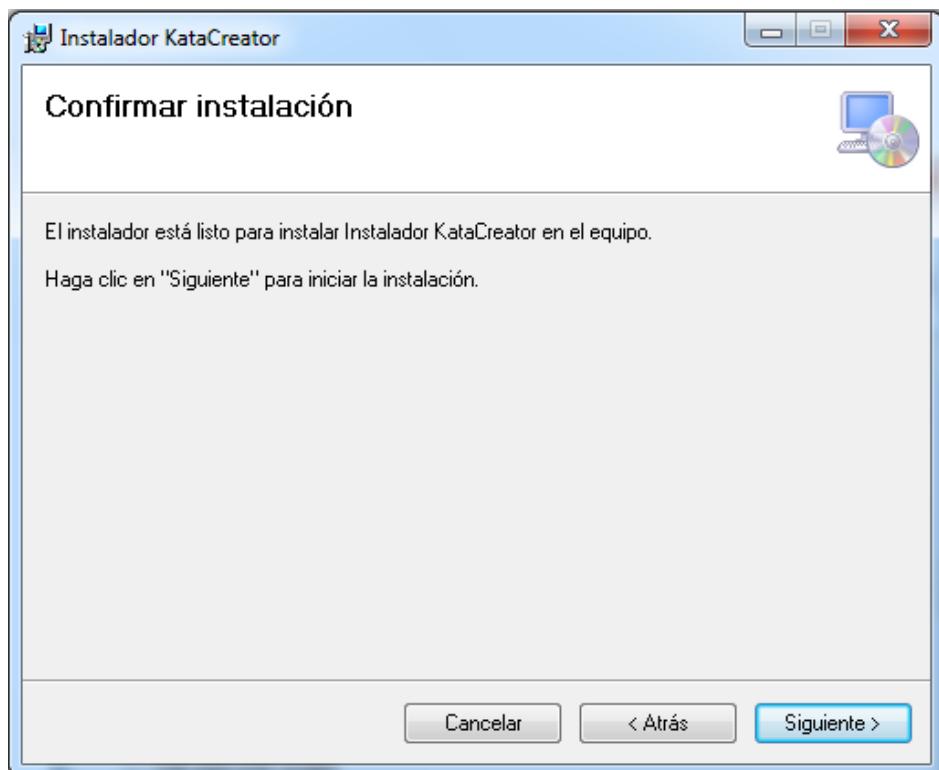


Figura 8.4: Confirma instalación en el equipo.

5. Selecciona Cerrar para finalizar instalación.

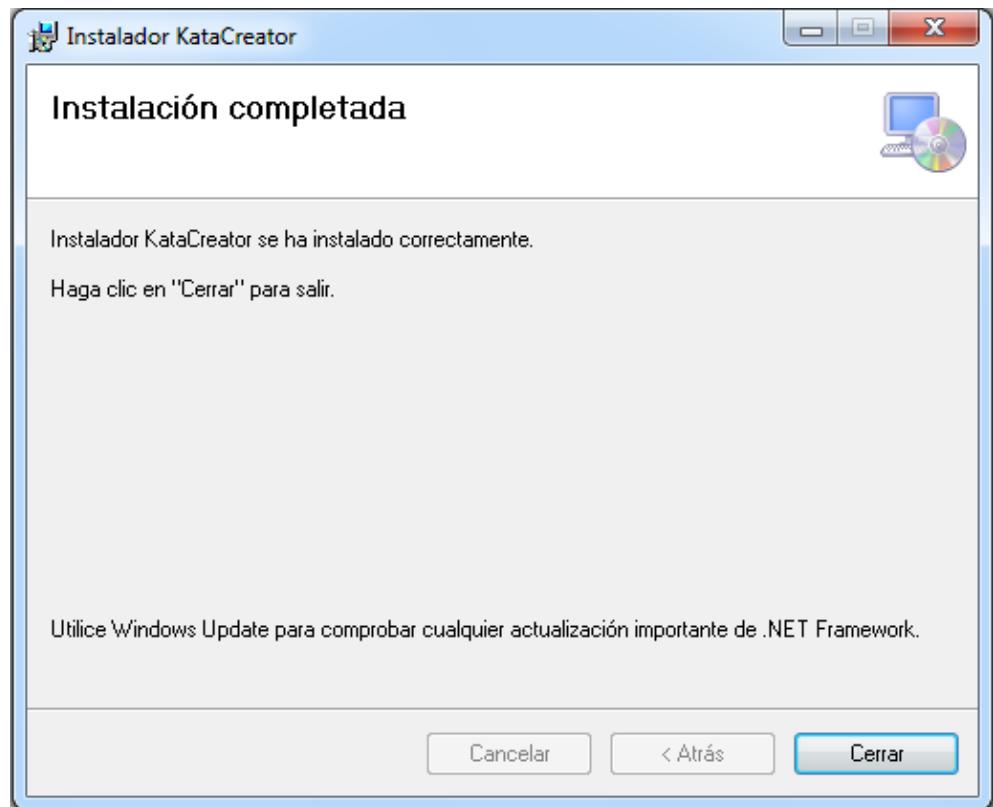


Figura 8.5: Cierra para salir.

### 8.2.2 Abrir/ejecutar KataCreator

Seleccione KataCreator.exe desde la carpeta donde se ha instalado *KataCreator* durante la instalación.

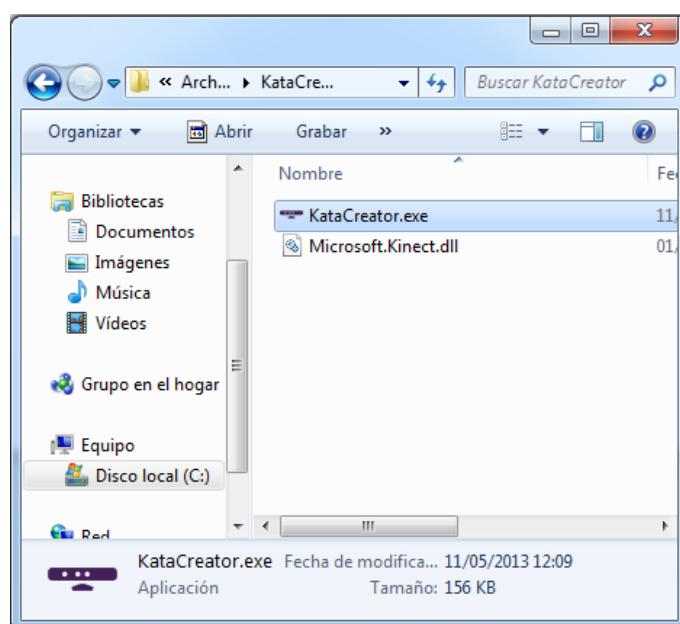


Figura 8.6: Ejecutar KataCreator.

### 8.2.3 Ventana principal de KataCreator

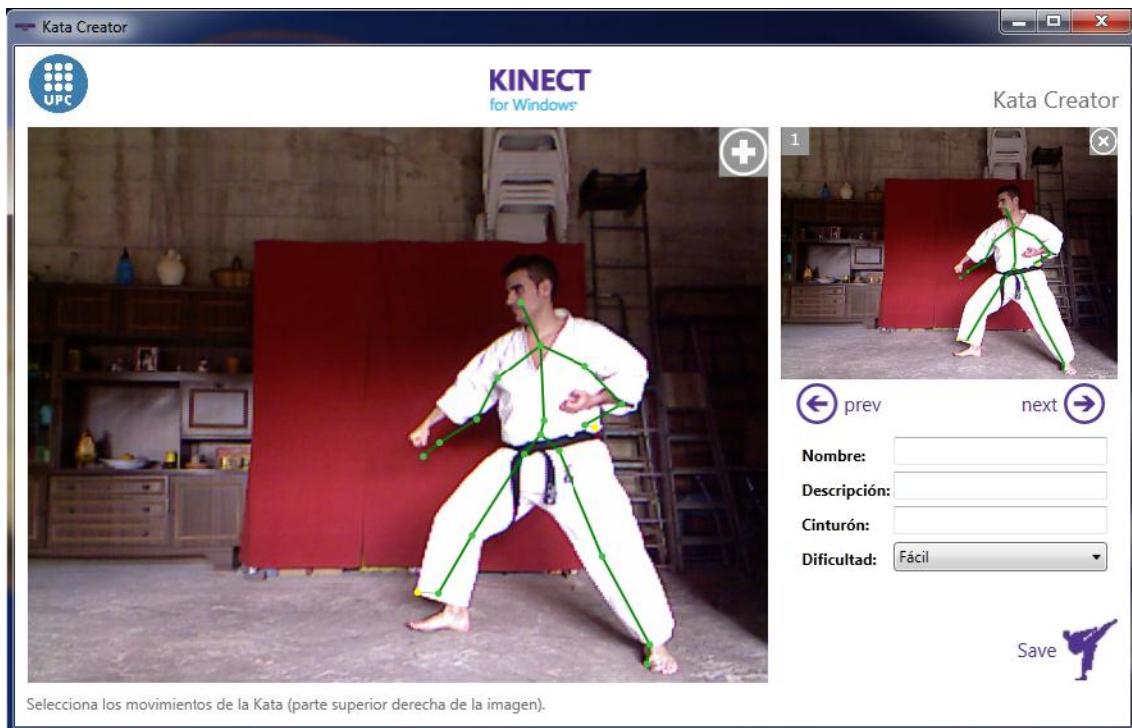


Figura 8.7: Vista principal de KataCreator.

A continuación se comentan todos los elementos de esta vista.

- 1. Logo y nombre de la aplicación:** Elementos que muestran el nombre de la aplicación y diferentes logos.

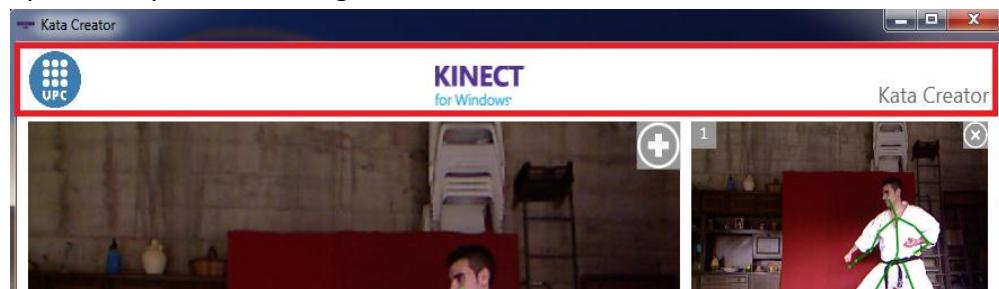


Figura 8.8: Nombre de aplicación y logos.

- 2. Imagen que recoge la Kinect:** Muestra al usuario la imagen actual que recoge la Kinect. En caso de que la Kinect no esté conectada al PC se muestra una imagen negra.



Figura 8.9: Imagen en tiempo real que recoge la Kinect.

- 3. Botón para añadir la postura actual que recoge la Kinect a la Kata:** Botón que sirve para crear la Kata. Permite añadir las posturas de las que se compone una Kata. En caso de que la Kinect aún no haya reconocido a un usuario activo nos informara mediante un pop-up.

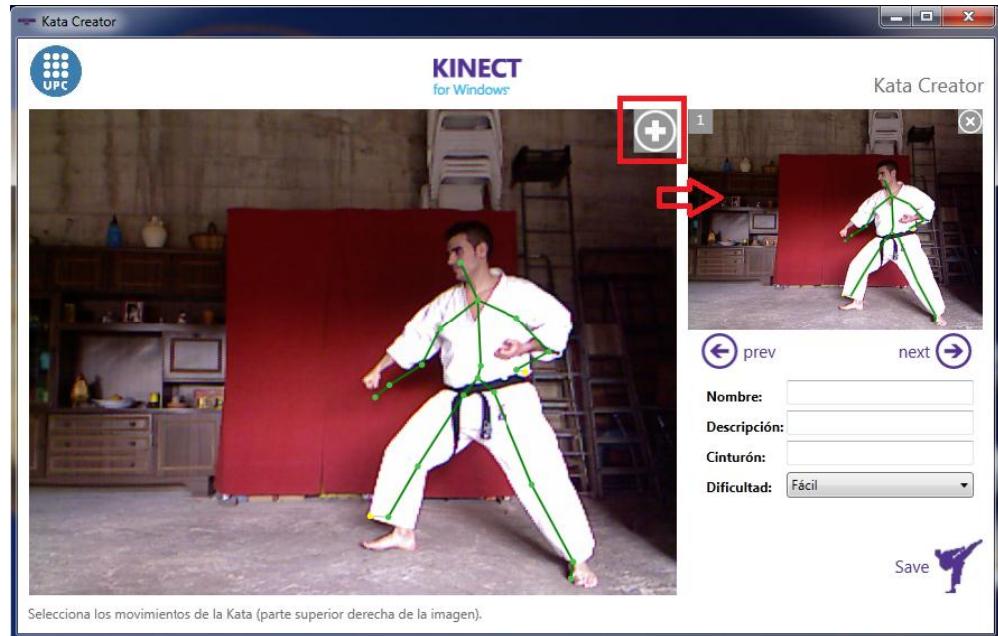
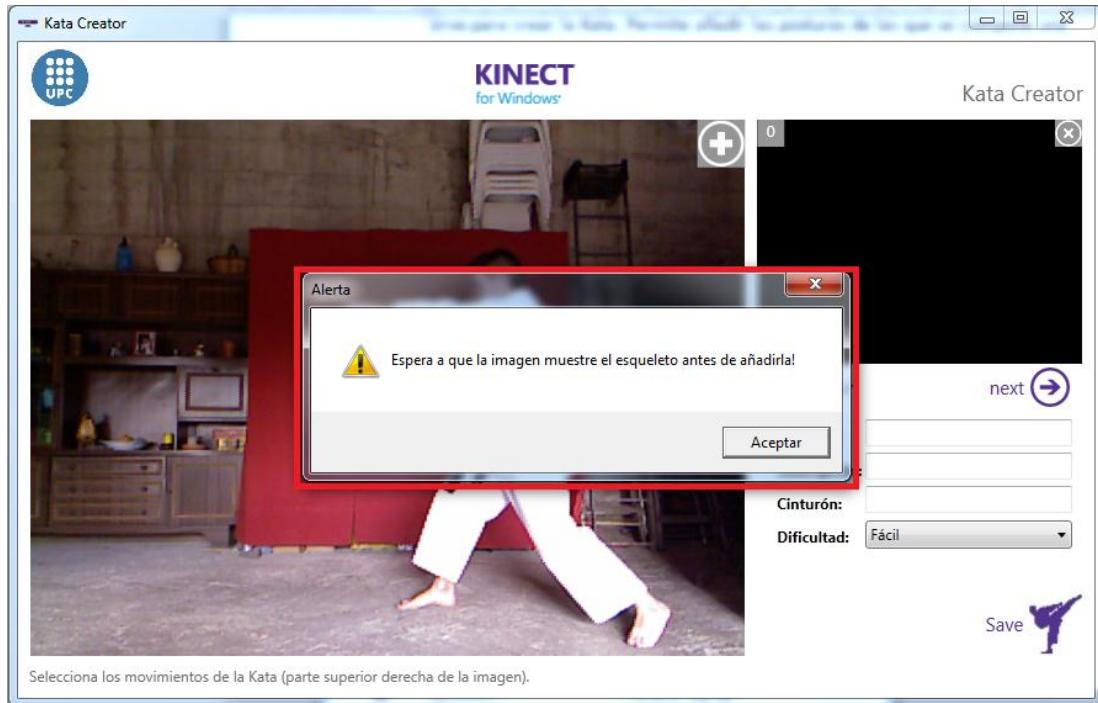


Figura 8.10: Botón para añadir postura a la Kata.

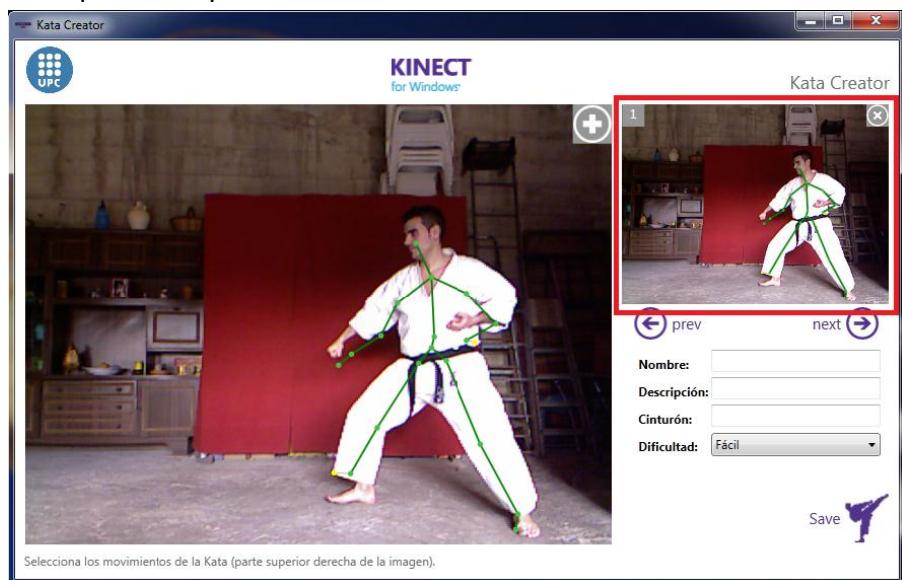
Si la Kinect aún no ha reconocido al usuario (no muestra el *Skeleton* de color verde en la imagen) no permite añadir la postura en la Kata y nos muestra el siguiente pop-up.



**Figura 8.11: Pop-up si de advertencia si intentamos añadir una postura cuando la Kinect aún no ha reconocido al usuario.**

#### 4. Imagen de la postura en la que nos encontramos de las que forman la Kata:

Esta imagen recoge una postura de las que hemos añadido en la kata. Al añadir una nueva postura se añadirá como la siguiente. Se nos permite navegar por las diferentes posturas que forman una kata.



**Figura 8.12: Postura en la que nos encontramos de las que forman la kata.**

- 5. Indicador de la posición de la postura en la Kata:** Nos muestra el número de la posición en la kata de la postura que se muestra al usuario.

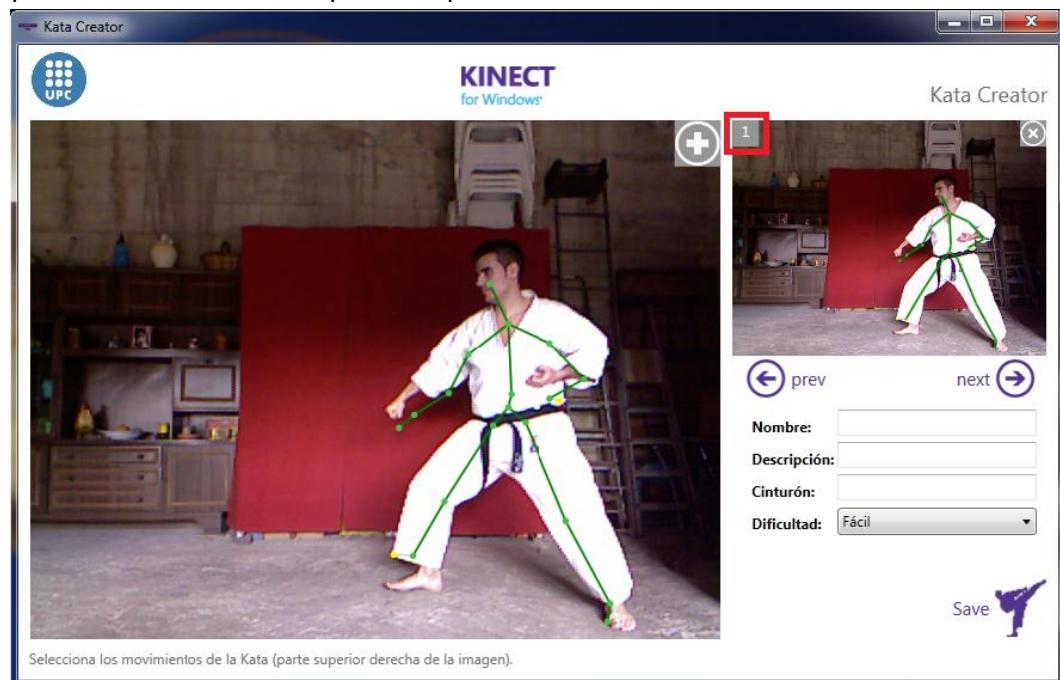


Figura 8.13: Posición de la postura de la kata.

- 6. Botón para eliminar la postura de la kata:** Botón que nos permite eliminar la postura de la kata que se muestra al usuario. Si no hay ninguna postura añadida en la kata no hace nada.

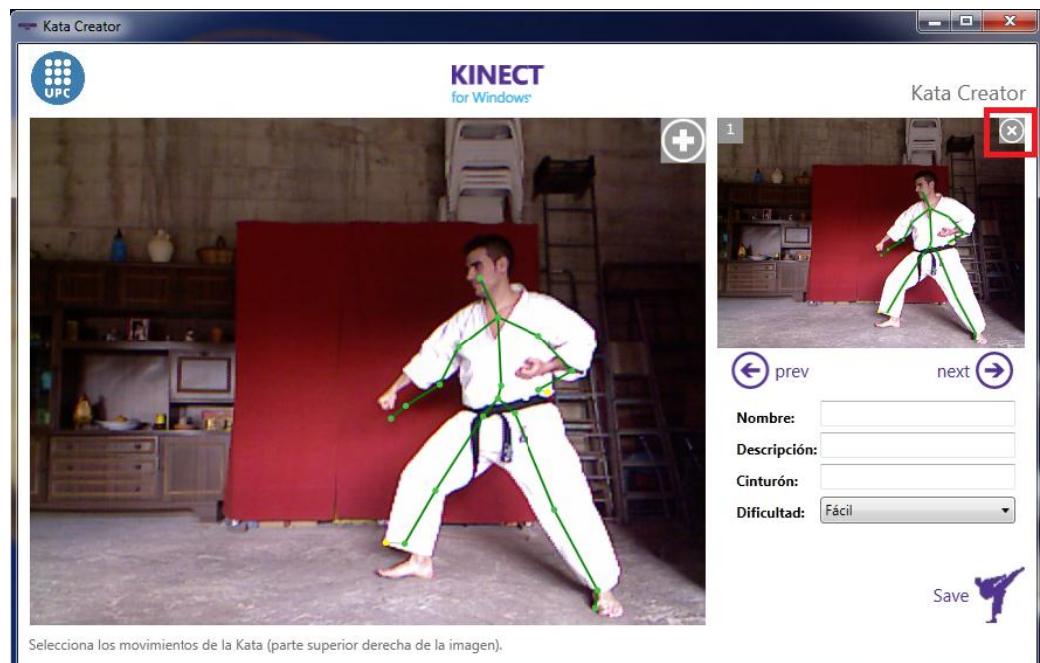


Figura 8.14: Elimina postura que se muestra.

- 7. Botones para navegar por la kata:** Botones que permiten avanzar o retroceder por las posturas que forman la kata.

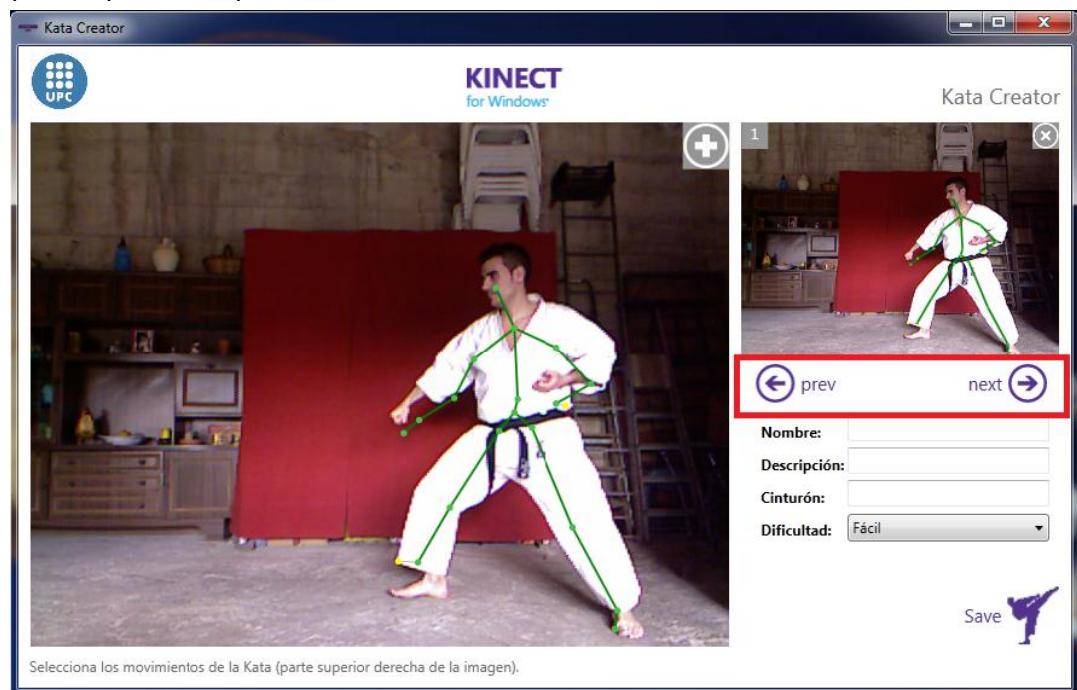


Figura 8.15: Botones que permiten navegar entre las posturas de las katas.

- 8. Información de la Kata:** Información de la Kata. Nombre, descripción, cinturón y dificultad que el usuario tiene que llenar. Esta información es necesaria para guardar la kata.

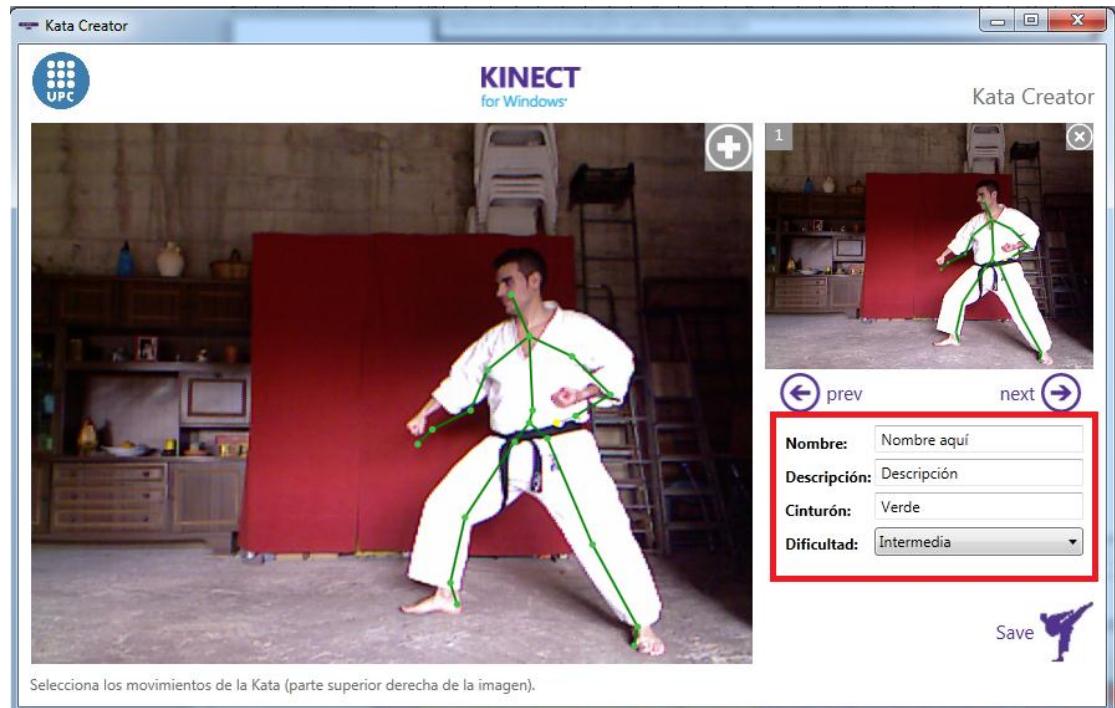


Figura 8.16: Información de la kata.

**9. Texto de información y ayuda de la aplicación:** Texto que nos indica como añadir posturas en la kata. En el caso de que la Kinect no este conectada es este mismo texto el que nos informa.



Figura 8.17: Información y ayuda para el usuario.

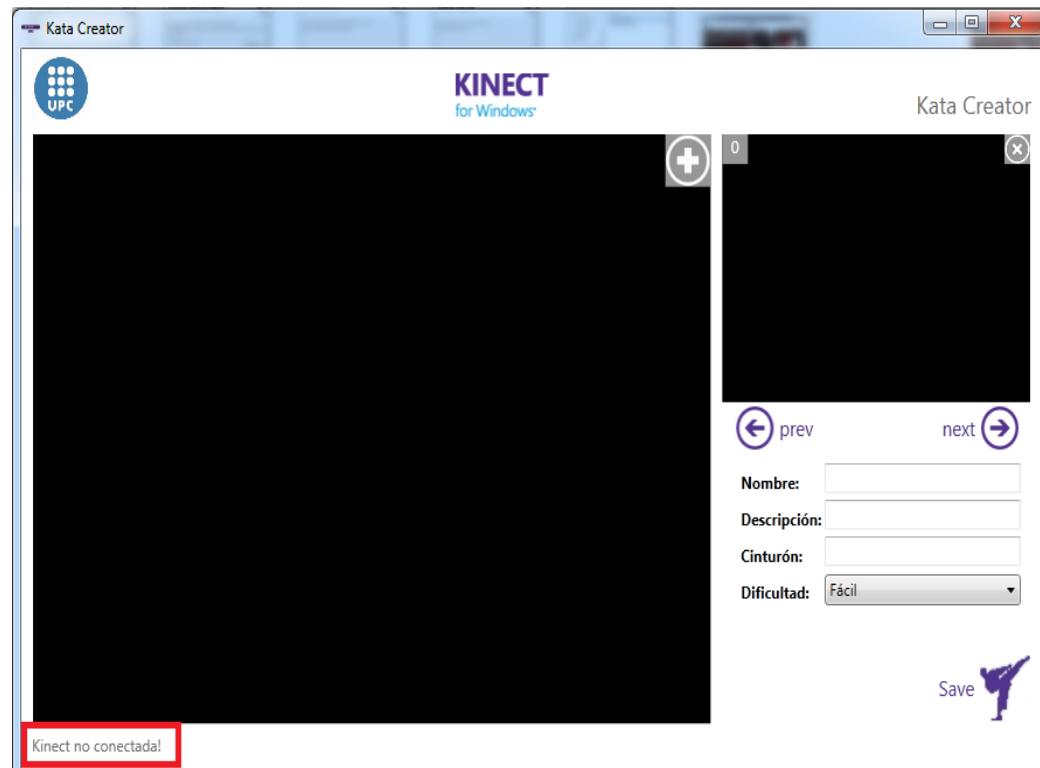


Figura 8.18: Kinect no conectada, información para que el usuario conecte la Kinect.

**10. Botón para guardar Kata:** Botón que permite guardar la kata creada en disco para su uso en *KataTraining*. Para que *KataTraining* disponga de la kata tiene que guardarse en la carpeta “Katas” de la ubicación de instalación de *KataTraining*. El botón de “save” de la kata no funciona si no esta toda la información o no hay ninguna postura que la forma.

La imagen que mostrará *KataTraining* al listar las katas es la se muestra en la imagen de la postura en la kata.

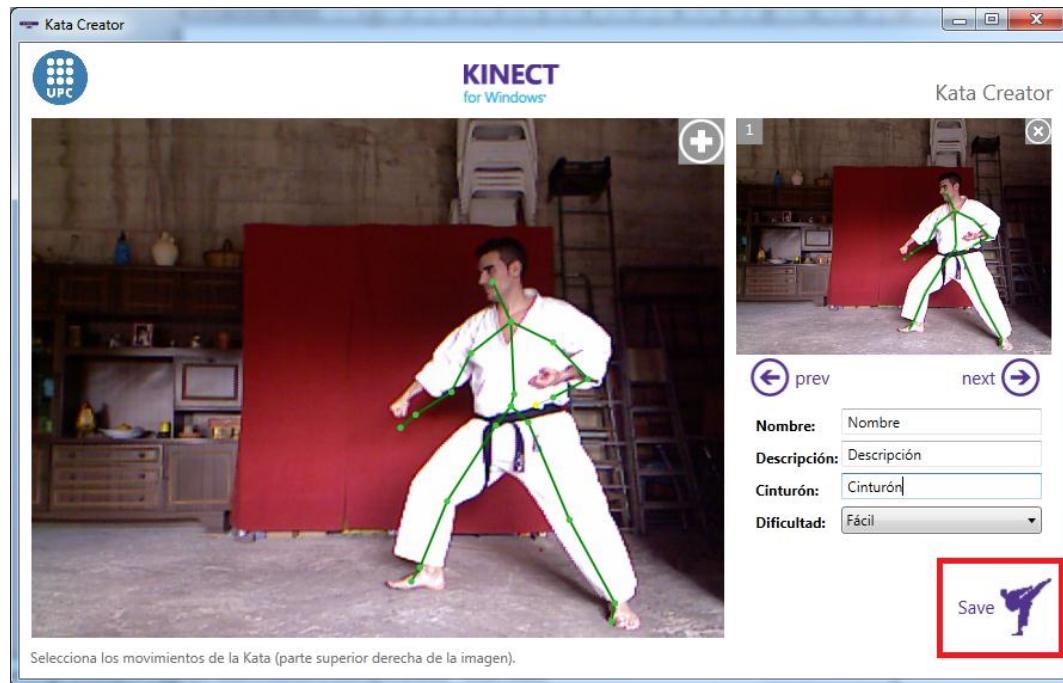


Figura 8.19: Guardar la Kata creada.

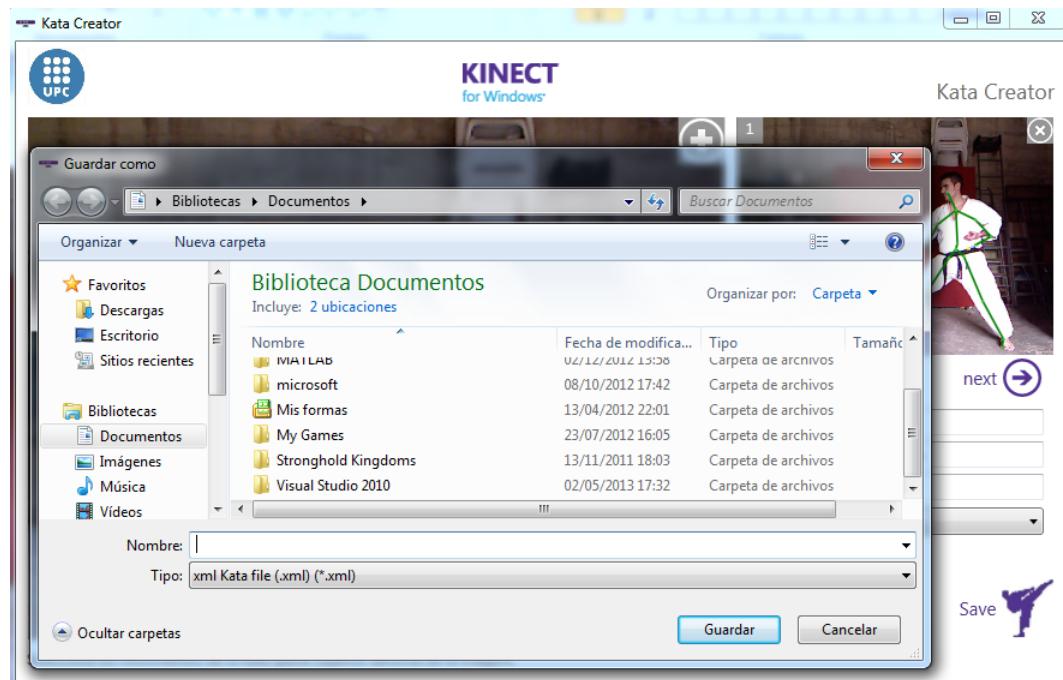
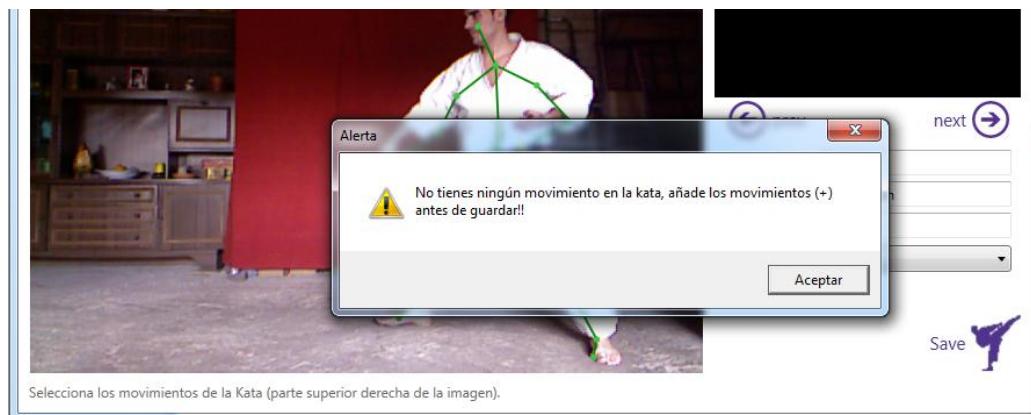


Figura 8.20: Elegir ruta para guardar la kata.

Si la kata no tiene posturas no se guarda nada en disco y muestra el siguiente pop-up para informar al usuario.



**Figura 8.21:** Pop-up al intentar guardar una kata sin ninguna postura.

Si la kata no tiene toda la información (nombre, descripción, cinturón y dificultad) no se guarda nada y muestra el siguiente pop-up para informar al usuario.



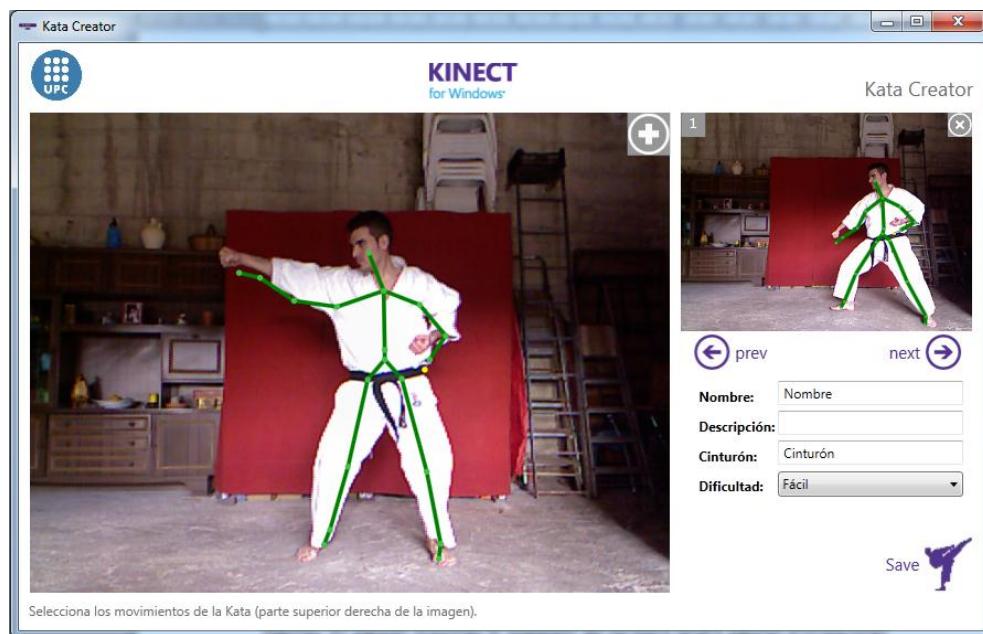
**Figura 8.22:** Pop-up al intentar guardar sin toda la información de la kata.

#### 8.2.4 Observaciones y recomendaciones para la creación de una kata

A la hora de crear una kata con *KataCreator* hay que tener en cuenta varios aspectos que vamos a comentar aquí.

En primer lugar, la Kinect funciona correctamente hasta los 3.5-4 metros, por lo que la persona que este haciendo la kata frente a la Kinect no puede sobrepasar esta distancia. Por otro lado es importante en Karate que la Kinect capte todo el cuerpo, por lo que **la distancia ideal para realizar la kata es alrededor de 3 metros enfrente de la Kinect**. Esto excluirá algunas katas en las que el karateca tenga de desplazarse más del rango permitido (esto se ve en la imagen que recoge *KataCreator* en tiempo real de la Kinect).

Por otro lado, como se ve en las imágenes anteriores de *KataCreator*, la aplicación muestra el esqueleto del karateca de color verde. Si no se muestra nada es que Kinect aún no ha reconocido a ningún usuario y hay que esperar hasta que lo muestre. Además, en algunas ocasiones el esqueleto del karateca tendrá algunas articulaciones en amarillo, esto indica que la Kinect tiene problemas para reconocer la postura actual y puede hacerlo mal. Por este motivo, **es muy recomendable que se evite añadir posturas en la Kata cuando haya varios elementos del esqueleto de color amarillo.**



**Figura 8.23: Postura buena para añadir en la kata.**



**Figura 8.24: Postura mal captada por Kinect, evitar a toda costa para resultados satisfactorios.**

### 8.2.5 Desinstalación de KataCreator

Hay dos formas de desinstalar *KataCreator* del sistema, ejecutando el instalador y seleccionando la opción de desinstalar o bien en Panel de **Control > Programas > Programas y Características**, selecciona *KataCreator* y clic derecho → desinstalar.

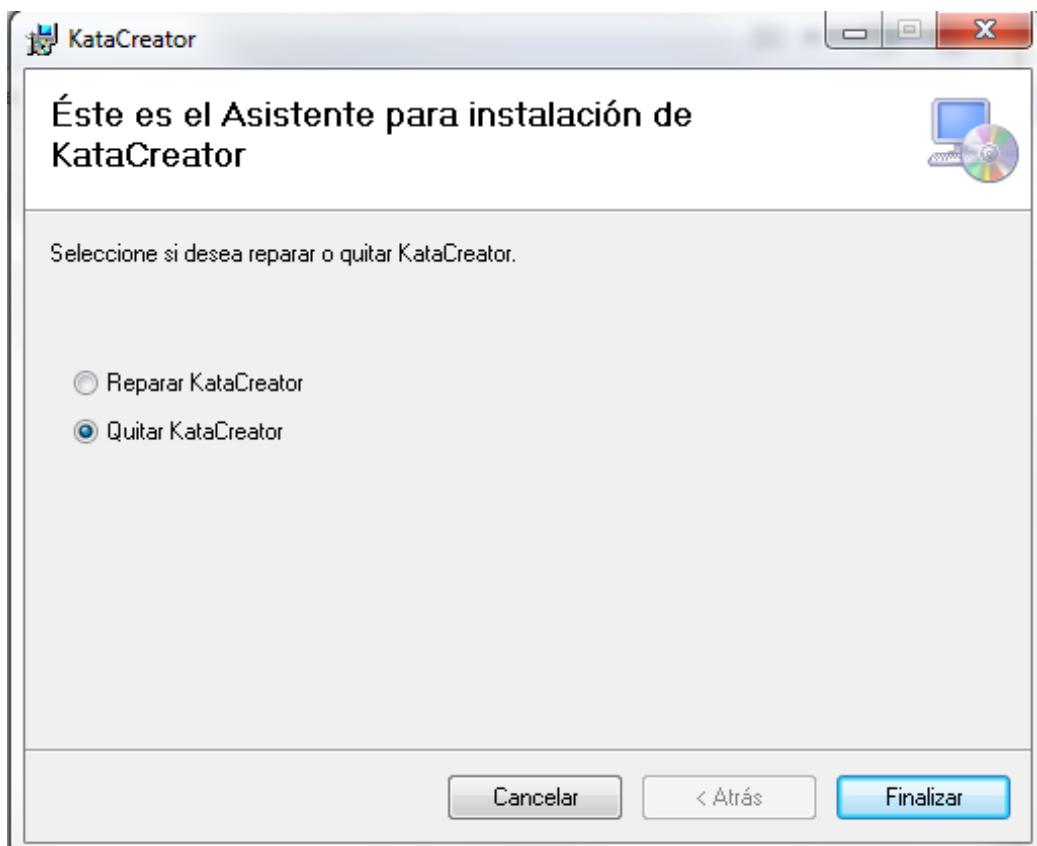


Figura 8.25: Desinstalando KataCreator.

## 8.3 Manual de KataTraining

### 8.3.1 Instalación

Antes de poder empezar a usar *KataTrainig* se requiere pasar por un sencillo proceso de instalación. Vamos a detallar el proceso paso por paso.

1. Ir a la carpeta donde se encuentra el instalador de *KataTraining* y ejecutar *KataTraining2.msi*

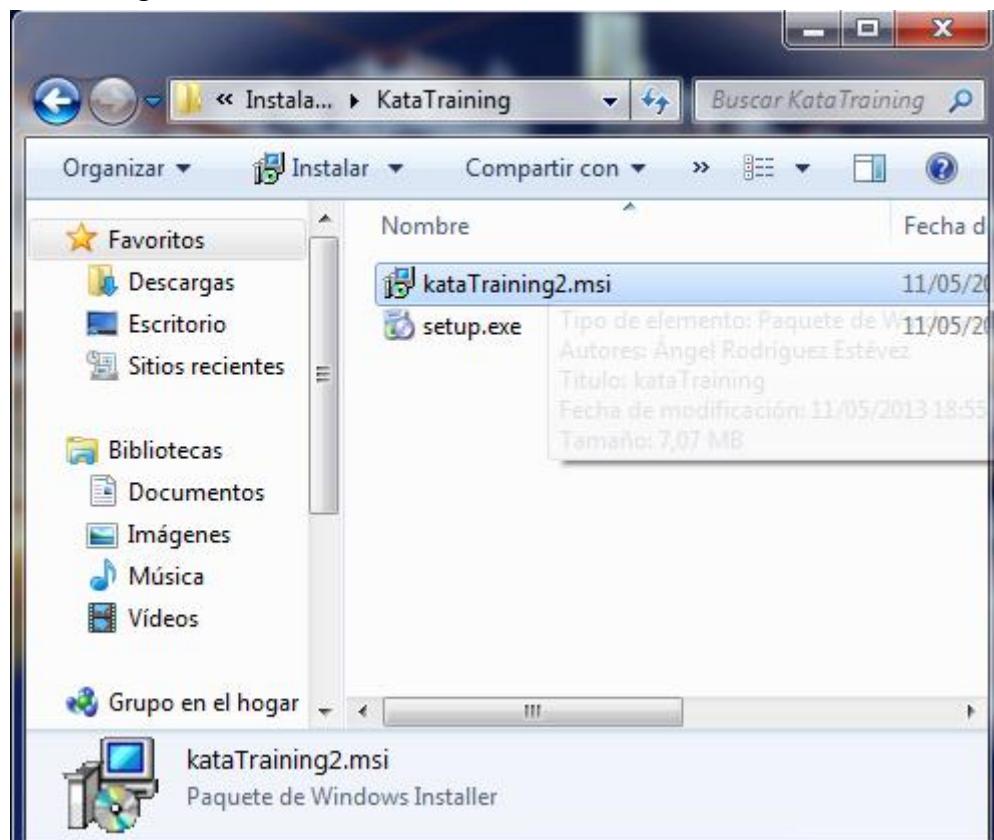


Figura 8.26: Ejecutar Instalador KataTraining.msi

2. Se abrirá el asistente de instalación, seleccione siguiente.

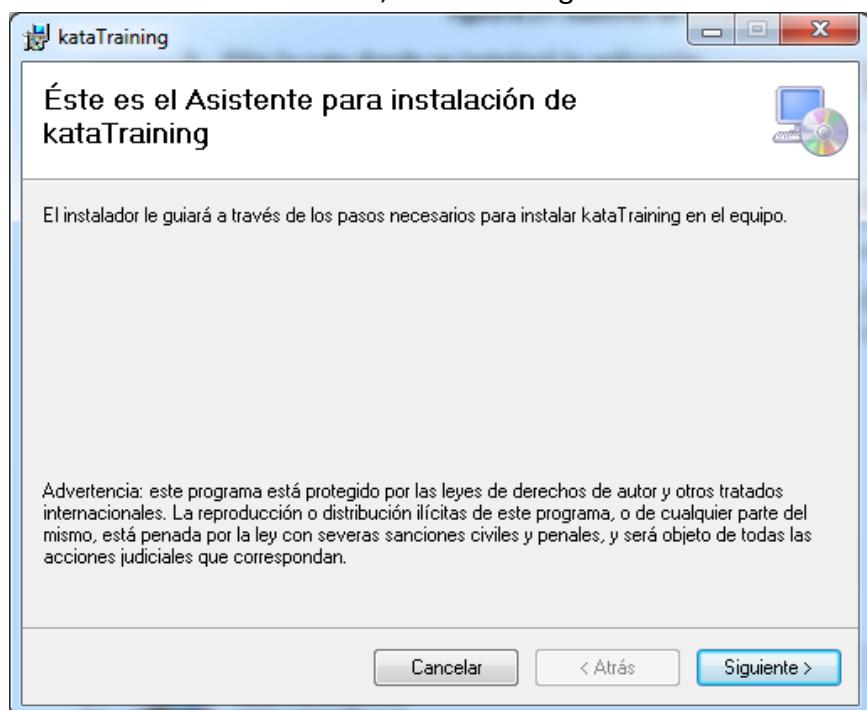


Figura 8.27: Asistente de instalación.

3. Elija la ruta donde se instalará la aplicación.

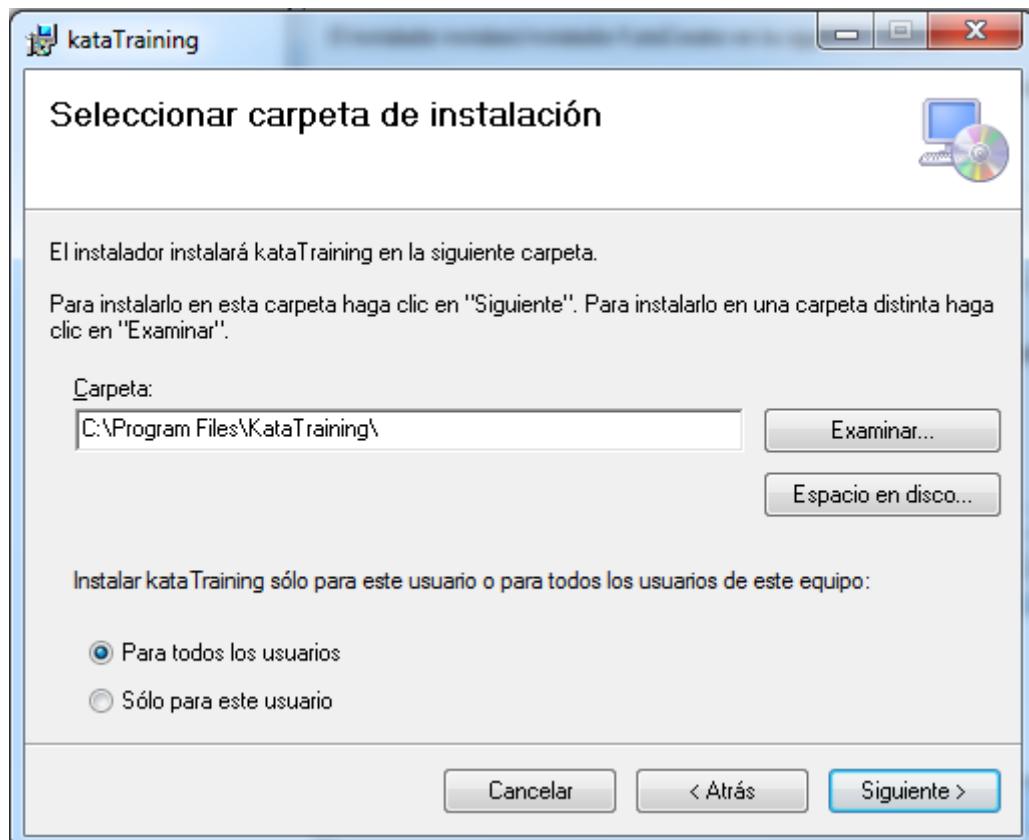


Figura 8.28: Seleccione ruta de instalación.

4. Confirma la instalación.

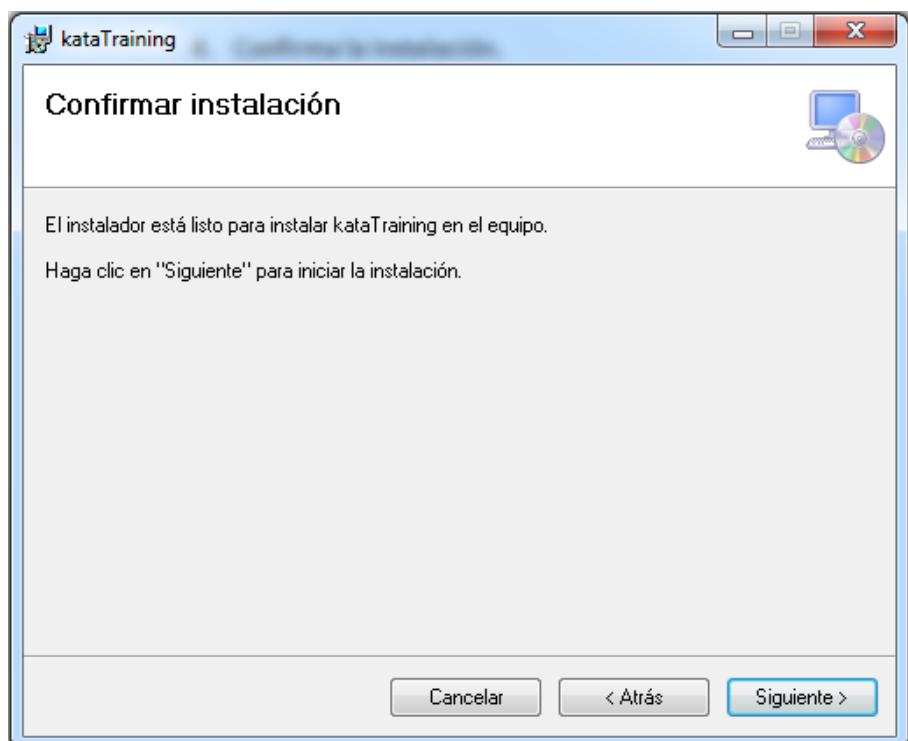


Figura 8.4: Confirma instalación en el equipo.

5. Selecciona Cerrar para finalizar instalación.

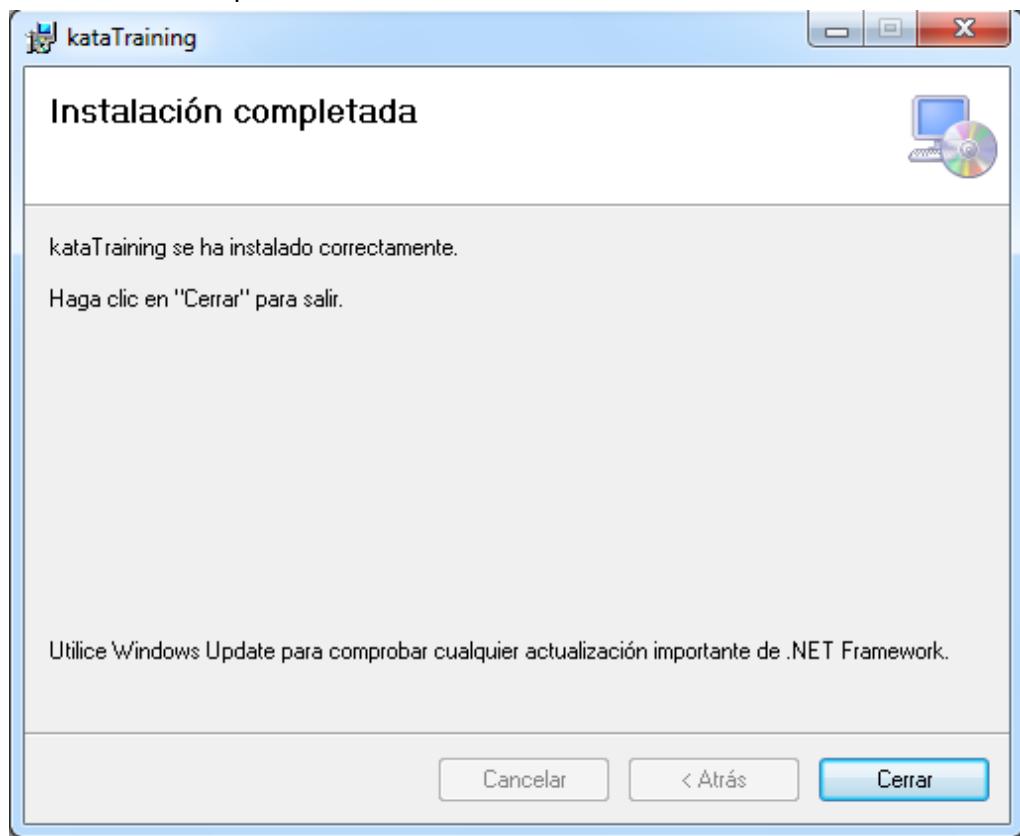


Figura 8.30: Cierra para salir.

### 8.3.2 Abrir/ejecutar KataTraining

Seleccione KataTraining.exe desde la carpeta donde se ha instalado *KataTraining* durante su instalación.

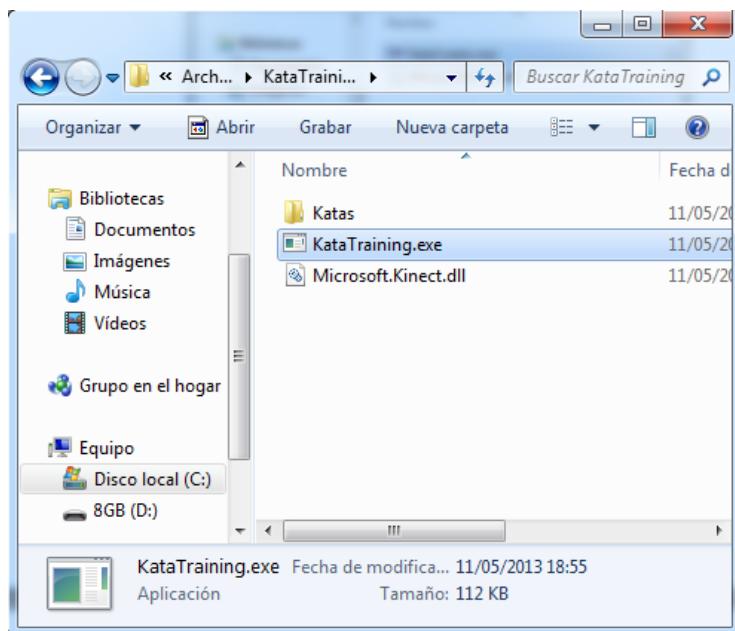


Figura 8.31: Ejecutar KataTraining.

### 8.3.3 Ventana principal de KataTraining

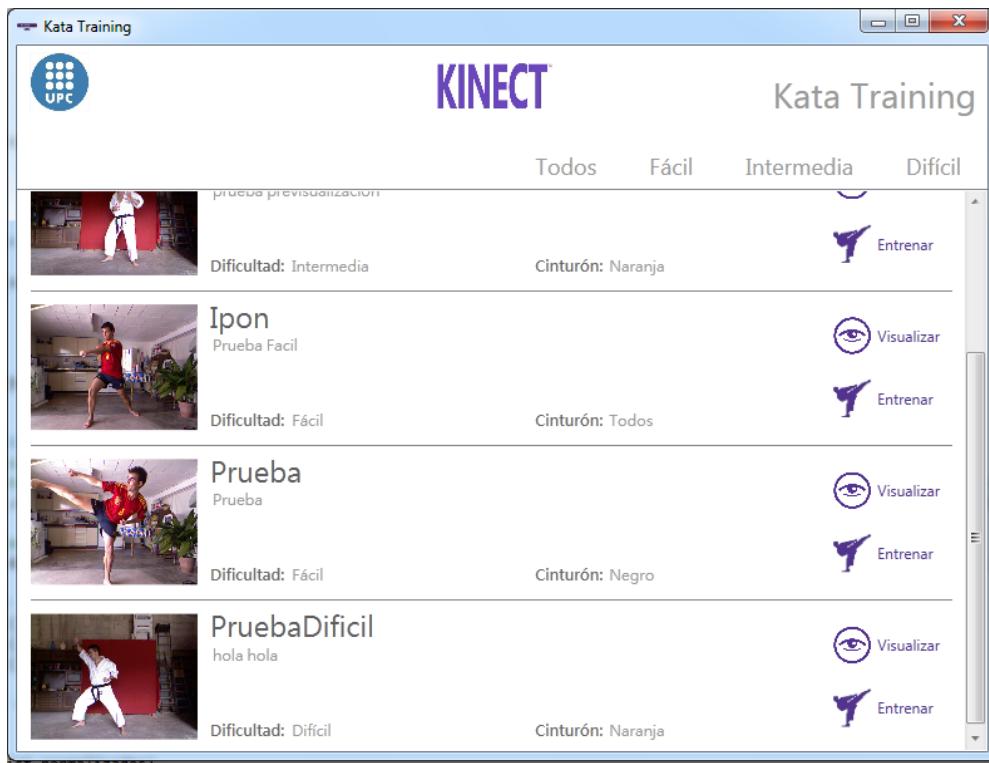


Figura 8.32: Vista principal de KataTraining.

A continuación se comentan todos los elementos de esta vista.

- Logo y nombre de la aplicación:** Elementos que muestran el nombre de la aplicación y diferentes logos.

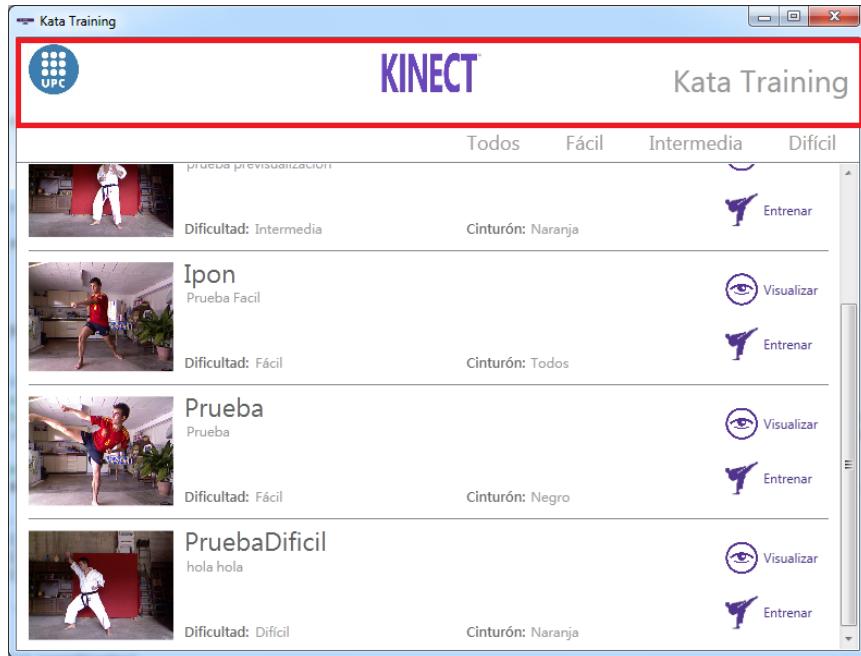


Figura 8.33: Nombre de aplicación y logos.

- 2. Botones para filtrar katas por dificultad:** Una serie de botones que sirven para seleccionar, de entre todas las katas que están en la carpeta “katas”, aquellas katas que se correspondan con la dificultad seleccionada.

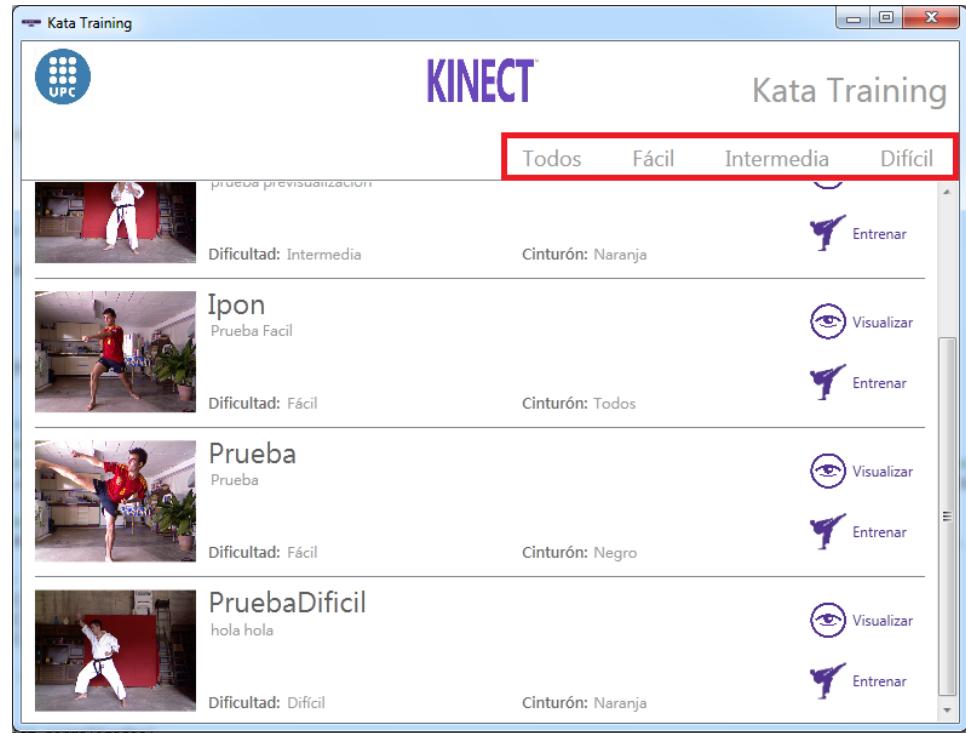


Figura 8.34: Botones para filtrar katas por dificultad.

- 3. Lista de Katas disponibles:** Se muestran al usuario todas las katas que están en la dentro de la carpeta “Katas”, de la misma ubicación que el ejecutable.

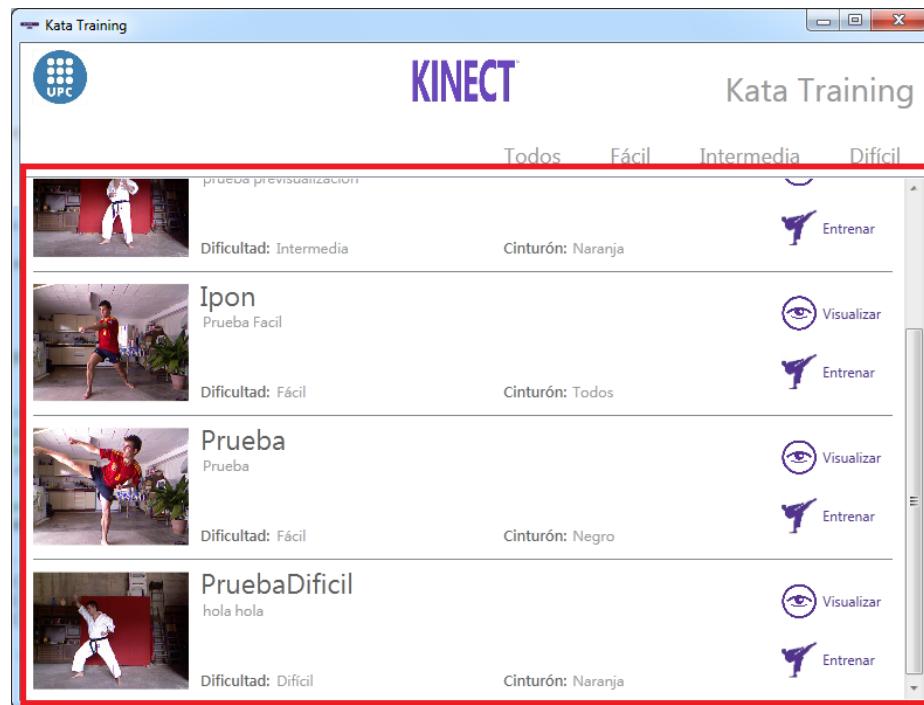


Figura 8.35: Lista de katas para visualizar/entrenar.

4. Botones de visualizar/entrenar una kata: Dos botones, uno de visualizar y otro de entrenar para cada kata de la lista de katas.

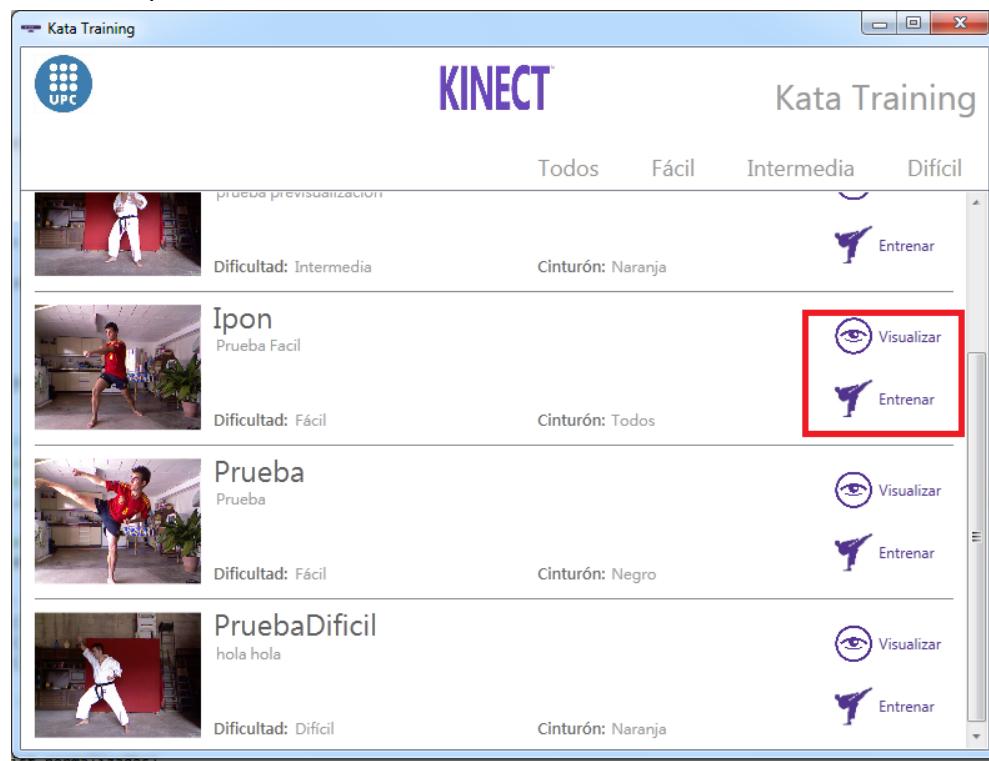


Figura 8.36: Botones de visualizar/entrenar una kata.

### 8.3.4 Ventana de Visualización de una kata

Cuando un usuario clica el botón de visualizar una kata nos muestra una vista que contiene el nombre de la kata y una imagen que nos muestra las posturas de la kata. Esta imagen avanza cada pocos segundos y se nos indica el número de posición de la postura dentro de la kata.

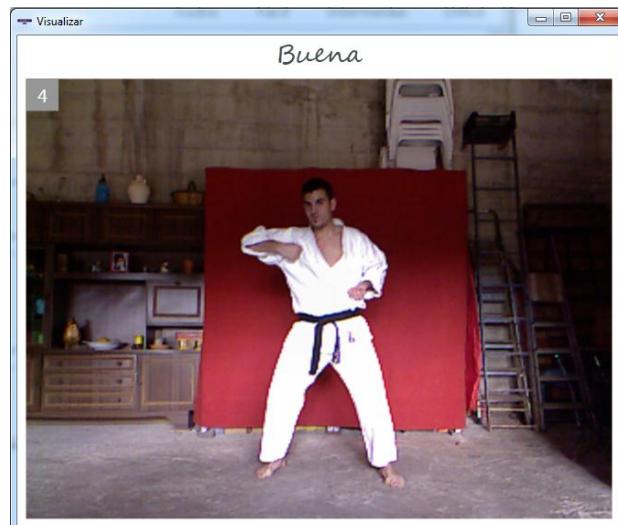


Figura 8.37: Visualización de una kata. Muestra la postura 4.

### 8.3.5 Ventana de Entrenamiento de una kata

Cuando un usuario clica el botón de entrenar una kata nos muestra una vista que nos permite practicar esa kata.

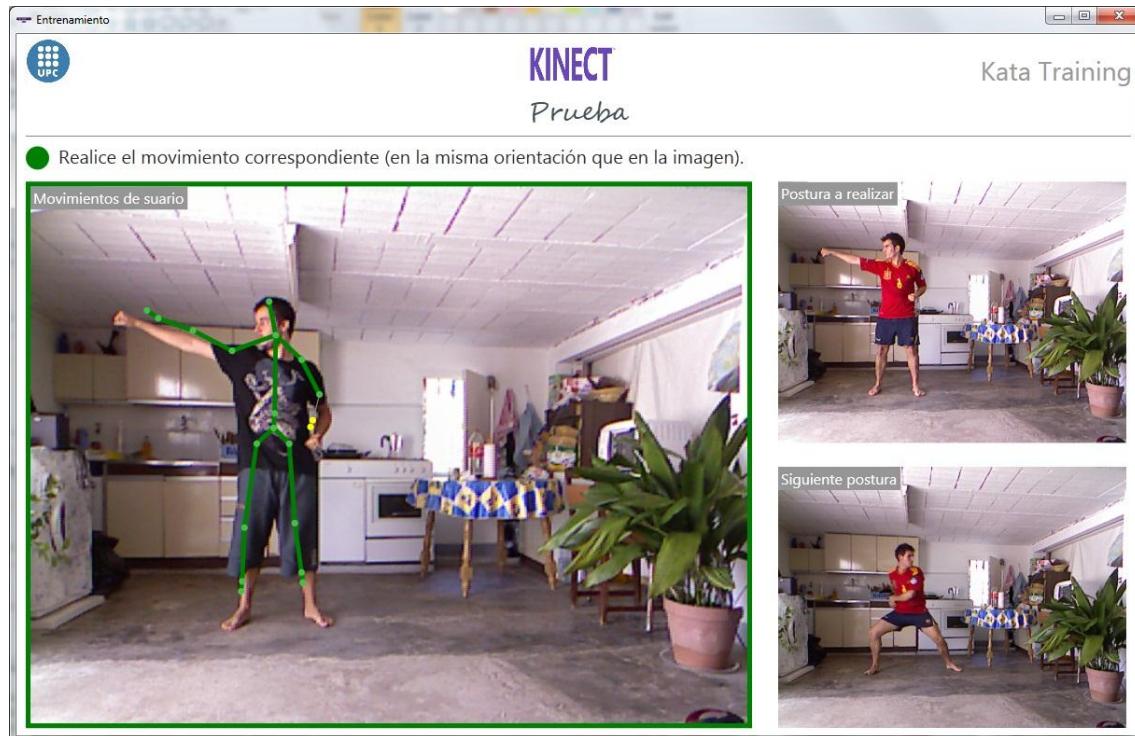


Figura 8.38: Vista de entrenamiento de una kata.

A continuación se comentan todos los elementos de esta vista.

- 1. Logo, nombre de la aplicación y nombre de la kata:** Elementos que muestran el nombre de la aplicación, diferentes logos y el nombre de la kata a entrenar.

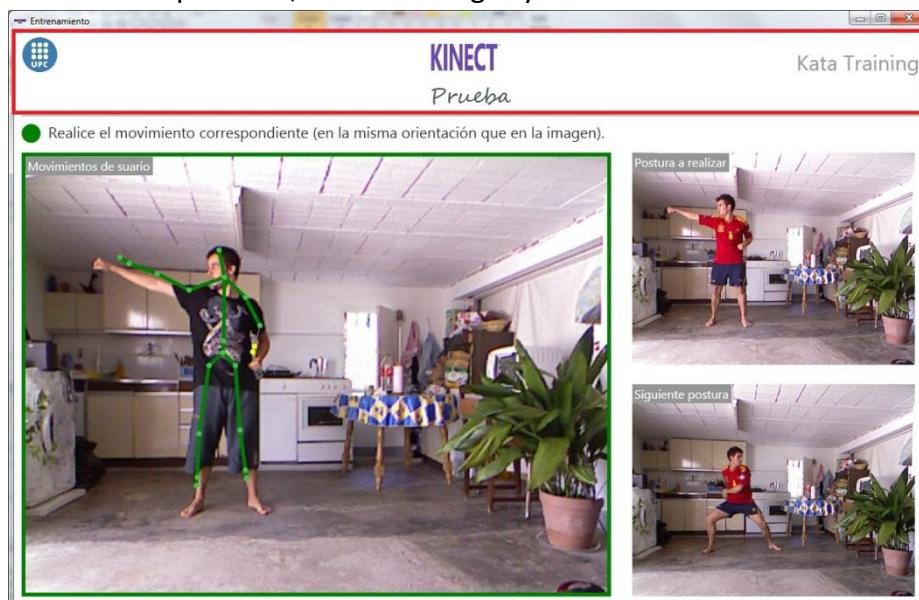


Figura 8.39: Nombre aplicación, logos y nombre de la kata.

- 2. Información para el usuario:** Texto que indica al usuario lo que tiene que hacer. Si la kinect no está conectada le indica que la conecte, si la kinect aún no ha reconocido al usuario le pide que espera antes de empezar. Una vez que el usuario puede empezar a realizar la kata le indica que haga el movimiento correspondiente.



Figura 8.40: Información para usuario.

 Asegúrese de que la kinect esta conectada correctamente.

Figura 8.41: Mensaje si la Kinect no está conectada al pc.

- 3. Imagen de usuario:** Imagen del usuario que está haciendo la kata enfrente de la Kinect. El contorno verde indica que el usuario ha hecho correctamente la postura que tiene que hacer.

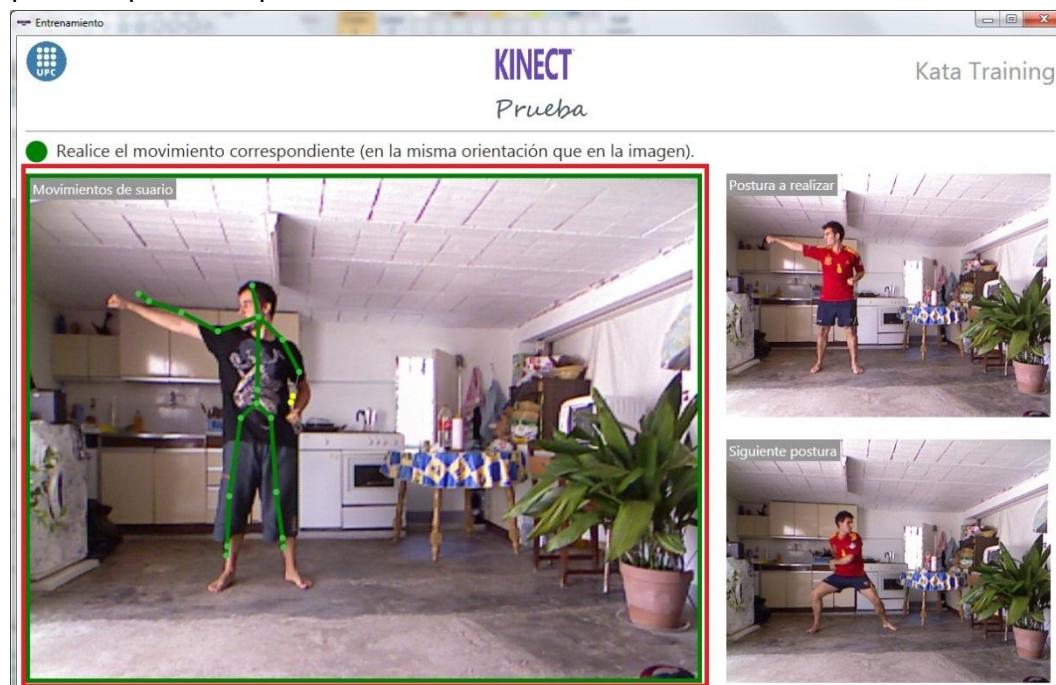


Figura 8.42: Usuario realizando postura de la kata.

- 4. Imágenes que muestran las dos siguientes posturas a realizar:** Dos imágenes que muestran las dos siguientes posturas a realizar por parte del usuario. La de arriba es la postura que tiene que realizar el usuario y la de abajo es la siguiente. En caso de que el usuario realice la última postura de la kata la ventana se cierra para seleccionar otra kata.

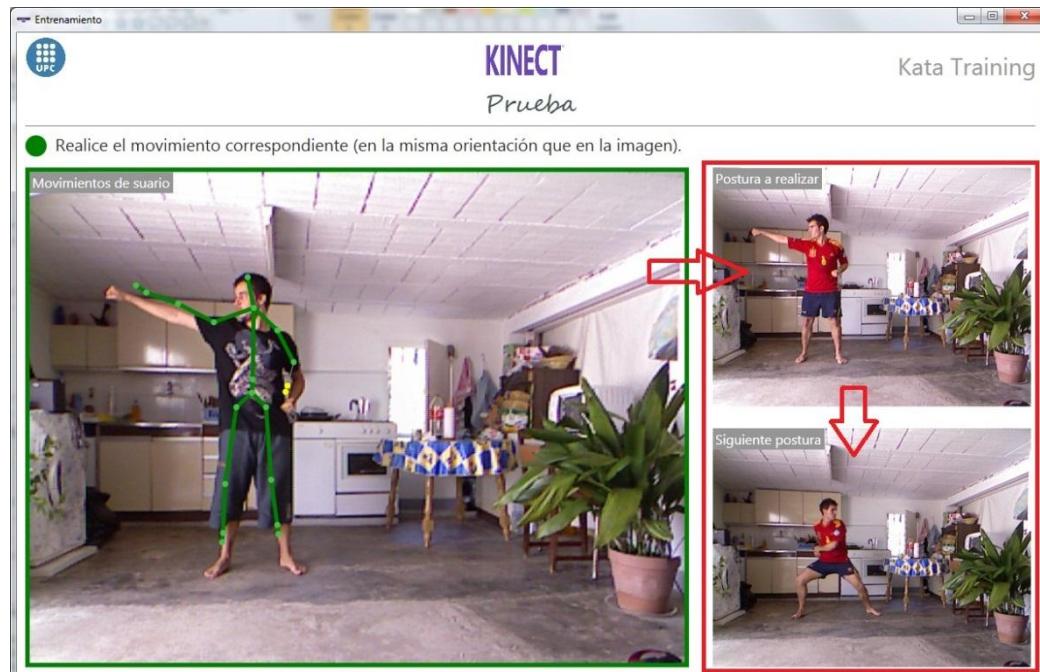


Figura 8.43: Dos posturas consecutivas que tiene que realizar el usuario.

Para que el usuario pueda entrenar las katas tiene que cumplir una serie de condiciones. La Kinect tiene que captar el cuerpo entero del usuario y no puede estar a más de 3.5 metros de ella. La imagen del usuario tiene que mostrar el esqueleto del usuario. El usuario tiene que seguir las instrucciones que están encima de la imagen del usuario.

### 8.3.6 Desinstalación de KataTraining

Hay dos formas de desinstalar *KataTraining* del sistema, ejecutando el instalador y seleccionando la opción de desinstalar o bien en Panel de **Control > Programas > Programas y Características**, selecciona *KataTraining* y clic derecho –>desinstalar.

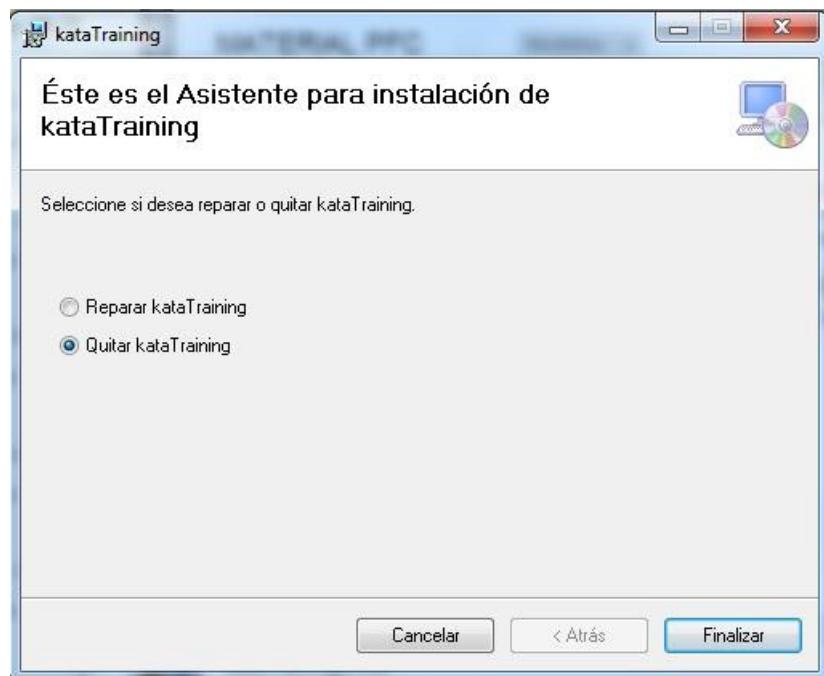


Figura 8.44: Desinstalando KataTraining.

## 9. Bibliografía

- [1] Motion Capture. [https://en.wikipedia.org/wiki/Motion\\_capture](https://en.wikipedia.org/wiki/Motion_capture)
- [2] Primer dispositivo de motion capture. <http://es.wikipedia.org/wiki/Zoopraxiscopio>
- [3] Rotoscopia. <http://es.wikipedia.org/wiki/Rotoscopio>
- [4] Kinect para Windows. <http://www.microsoft.com/en-us/kinectforwindows/>
- [5] SDK oficial de Microsoft para Kinect. <http://msdn.microsoft.com/en-us/library/hh855347.aspx>
- [6] Driver LibFreenect. [http://openkinect.org/wiki/Main\\_Page](http://openkinect.org/wiki/Main_Page)
- [7] SDK OpenNi para Kinect. <http://www.openni.org/>
- [8] API SDK Kinect: KinectSensor. <http://msdn.microsoft.com/en-us/library/hh855413.aspx>
- [9] API SDK Kinect: ColorStream. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.kinectsensor.colorstream.aspx>
- [10] API SDK Kinect: DepthStream. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.kinectsensor.depthstream.aspx>
- [11] API SDK Kinect: SkeletonStream. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.kinectsensor.skeletonstream.aspx>
- [12] API SDK Kinect: Skeleton. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.skeleton.aspx>
- [13] API SDK Kinect: Joints. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.skeleton.joints.aspx>
- [14] KinectStudio. <http://msdn.microsoft.com/en-us/library/hh855389.aspx>
- [15] Windows Presentation Foundation. <http://msdn.microsoft.com/es-es/library/ms754130.aspx>

[16] XNA. <http://msdn.microsoft.com/en-us/aa937791>

[17] Test Driven Development. [http://en.wikipedia.org/wiki/Test-driven\\_development](http://en.wikipedia.org/wiki/Test-driven_development)

[18] Kinect voice. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.kinectsensor.audiosource.aspx>