RASMA: Plataforma para la Creación de Animaciones Stop-Motion con Asistencia Robótica

David Cuenca, Iván Perea, Gabriel J. García, Jorge Pomares, Fernando Torres
dacuetu@gmail.com, {ivan.perea, gjgg, jpomares, Fernando.Torres}@ua.es

Resumen

En este artículo se describe el concepto de plataforma RASMA, Robot-Assisted Stop-Motion Animation, cuya finalidd es facilitar la tarea de generar los fotogramas necesarios para crear una secuencia animada en 2D. Se describen tanto la generación de trayectorias que deben seguir los objetos (en Unity 3D o en Adobe Flash Player), como la exportación/importación de los ficheros de datos en XML, la planificación de las trayectorias del robot, la toma de fotogramas y el ensamblado final de toda la secuencia.

Palabras Clave: Planificación de trayectorias, stop-motion animation, animación cinemática.

1 INTRODUCCIÓN

La técnica de animación denominada Stop-motion consiste en la creación de una secuencia de vídeo donde se aparenta el movimiento de objetos estáticos a partir de una serie de imágenes fijas sucesivas. Aunque existen multitud de programas para la generación de animaciones Stop-motion tanto libres (linuxstopmotion.org, Luciole, etc) como comerciales (AnimatorHD, Dragonframe, StopMotionPro...), todos ellos parten de la premisa de que el animador es el que mueve manualmente la escena y activa el disparador de la cámara manualmente o cada cierto periodo de tiempo. Tras analizar las posibilidades que brindan estos programas, ninguno de ellos contempla la posibilidad de que un robot componga la escena y genere el vídeo de forma desatendida.

Actualmente hay pocas investigaciones que aúnen robótica y animación Stop-motion, pero sí en cambio animación Stop-motion y prototipado de interacciones tangibles [1]. Por otro lado, el juguete Topobo desarrollado en el MIT utiliza conceptos que pueden utilizarse en el campo de la animación Stop-motion, al hacer un pre-programado cinemático de un sistema modular con motores/encoders en las articulaciones [2]. Del campo de la fotografía time-lapse (aquella en la que se toman fotografías cada cierto tiempo), surgió Whizard Motion, un sistema para controlar la toma de fotogramas coordinando distintos ejes de posición de una cámara fotográfica [3]. Hay investigaciones en el campo del posicionamiento de objetos (pick & place) que podrían ser aplicables en la animación Stop-motion robótica, tal es el caso de [4], donde con una nube de puntos se elabora un modelo de un objeto tridimensional y se define la posición preferida.

En este artículo, se define una plataforma para elaborar trayectorias virtuales de objetos según un modelado físico, trasladar esas posiciones de los objetos a un escenario donde un brazo robótico es el encargado de realizar las iteraciones del movimiento y finalmente, capturar desde una cámara digital la escena una vez que los objetos están colocados según la posición precalculada para el fotograma y el brazo robótico está fuera del campo de visión.

En base a lo anterior definimos "RASMA" (Robot-Assisted Stop-Motion Animation) como aquella animación de tipo stop-motion en la que algunos o todos los elementos que componen la escena de un frame son colocados por un elemento automático de agarre y posicionamiento (pick&place).

Para generar este tipo de aplicaciones definimos una "plataforma de desarrollo RASMA" como el conjunto de herramientas (software y hardware) que permite animaciones RASMA. Incluye las siguientes partes: creador de bibliotecas de objetos, animador software de objetos, tracking software de posiciones, exportación de archivos de intercambio, importación de dichos archivos y de archivos con datos sobre la escena, software de manejo de un actuador automático de pick&place, captura automática de imágenes fijas (still shots), software de conversión de secuencia de imágenes fijas en video y, opcionalmente, mezclador de video y audio.

2 ESQUEMA DEL SISTEMA RASMA

En la Figura 1 se muestra el diagrama de bloques del sistema. El sistema RASMA se compone de:

1. Generación de trayectorias de los objetos: o bien en Adobe Flash, o bien en Unity 3D, se calculan las trayectorias de los objetos

teniendo en cuenta gravedad, colisiones, efectos físicos, etc. Esta información se exporta a un fichero XML.

2. Posicionamiento de los objetos en los frames reales: partiendo de la información generada anteriormente y con mediciones adicionales de los objetos, un programa que interactúa con el brazo robótico calcula las trayectorias pick&place del brazo robótico para cada fotograma.

3. Creación de video: Se ejecutan las instrucciones y, automáticamente, una cámara fotográfica va tomando imágenes según el software solicite (normalmente cuando el fotograma ha sido compuesto y el brazo robótico está fuera del área de visión de la cámara).

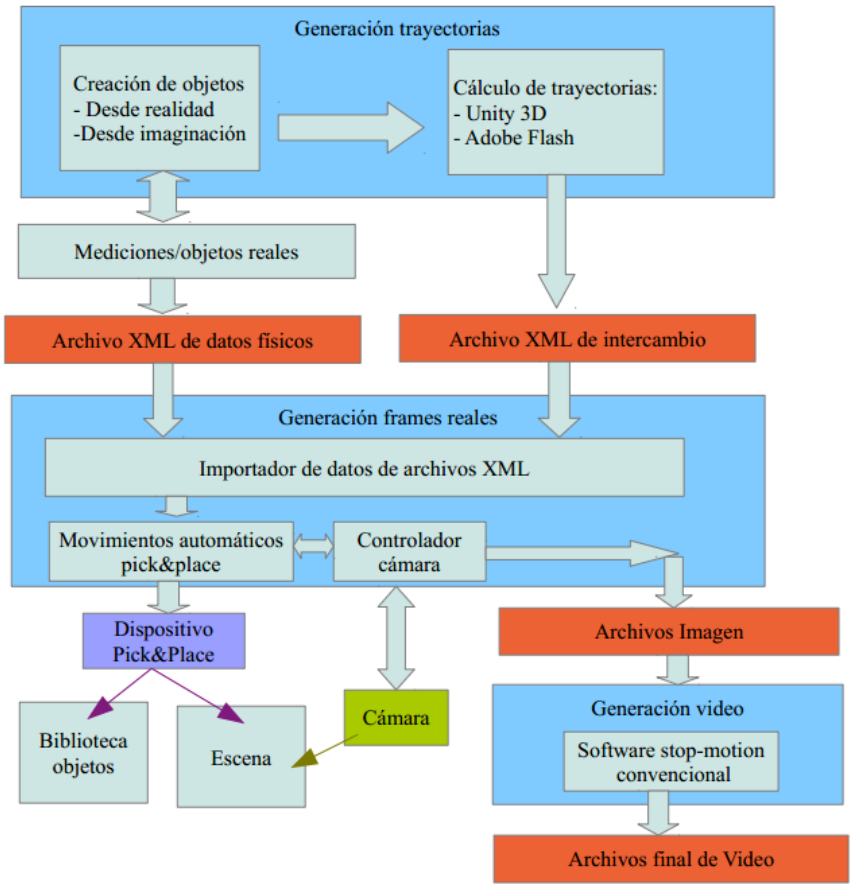

Figura 1: Diagrama de bloques

3 PLANIFICACIÓN DE TRAYECTORIAS

Para la generación de trayectorias se ofrecen dos posibilidades:

1. Unity 3D: originalmente utilizado para programación de videojuegos. La principal ventaja que proporciona su utilización es la aplicación de efectos físicos, tales como fuerzas, gravedad, par giratorio, etc.

2. Adobe Flash: originalmente para animaciones vectoriales. La principal ventaja que proporciona esta segunda alternativa es la animación según una trayectoria predefinida (spline). También tiene algunos componentes para aplicar efectos físicos.

En cualquiera de los supuestos se debe componer la escena según lo que queramos representar. Como ejemplo la Figura 2 muestra una escena compuesta en Unity 3D (que después se ha animado con un robot PA-10) y la Figura 3 otra escena compuesta en Adobe Flash (sólo como demostración).

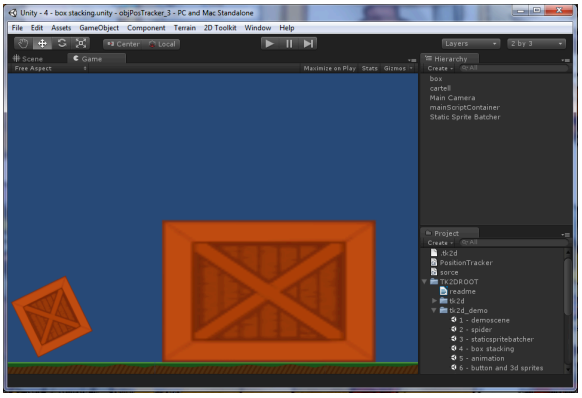

Figura 2: Escena compuesta en Unity 3D

3.1 ESCENA COMPUESTA CON UNITY 3D

En la vista 3D hay una cámara enfocando a los cuadrados. La vista que se ve en la imagen es la vista "de juego", es decir, cómo se ve la perspectiva desde esa cámara cuando el programa está ejecutándose. Las coordenadas que se debe tomar son las proporcionadas por esta cámara en píxeles y no las del sistema global.

En los otros paneles laterales se puede acceder a los distintos elementos del proyecto. Para componer esta escena en particular se ha usado como base la escena de prueba "box stacking", incluida en el Toolkit 2D. Dicho toolkit permite trabajar más fácilmente en 2D, tiene funciones específicas para cargar sprites y para manejar colisiones bidimiensionales. Otra posibilidad consiste en activar las propiedades "lock in axis Z", permitiendo trabajar en 3D puro simulando un entorno 2D al emplear una proyección ortogonal en la cámara.

Hay que tener en cuenta que cada objeto animado requiere tener un script asociado donde se definen los efectos cinemáticos o dinámicos aplicados al mismo. Además necesitamos un objeto adicional que contenga el script para hacer el tracking de posiciones de los demás objetos que estén marcados con el tag "trackedObject".

Para el funcionamiento del sistema de colisiones se requiere que cada GameObject en la escena tenga asociado un RigidBody. Un objeto RigidBody permite definir las propiedades físicas del objeto (peso, elasticidad, etc). Si no se asocia un RigidBody, el GameObject será incorpóreo, por lo que no colisionará con nada.

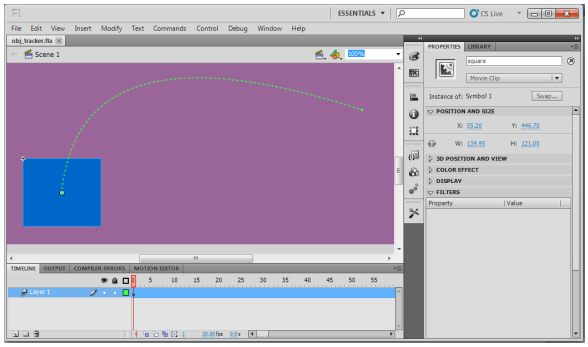

Figura 3: Escena compuesta en Unity 3D

3.2 ESCENA COMPUESTA CON FLASH

En vez de generar trayectorias a partir de efectos físicos (fuerzas, colisiones, etc), la composición de la escena mediante Adobe Flash requiere del uso de una técnica distinta. La llamada "interpolación de fotogramas clave" consiste en definir un fotograma clave inicial, y luego preparar un fotograma final arrastrando el objeto a su nueva posición. En Flash CS5 se consigue añadiendo "Motion Tween" al símbolo.

En la parte inferior de la pantalla se muestran los fotogramas clave y la duración de los "Motion tweens". Una vez generada la trayectoria, esta puede ser modificada tal como le interese al usuario.

Sin embargo, aunque dos objetos se crucen, actuarán como dos capas independientes la una de la otra a no ser que se programe algún tipo de interacción o se haga uso de los paquetes ofrecidos para tal efecto. Uno de tantos paquetes es Box2DFlashAS3, usado en el conocido juego Angry Birds.

Aunque Unity 3D se programa en C# o JavaScript y Adobe Flash en ActionScript3, un lenguaje similar a JavaScript, la estrategia en ambos casos es la misma:

1. Primero se compone una escena.
2. Se definen tamaños, trayectorias, choques, etc.
3. Se genera la escena y el script lo ejecutamos en cada frame para guardar la posición X,Y (en píxeles) de cada objeto y su orientación.
4. Finalmente se guardan todos los datos en un archivo XML.

3.3 XML DE ANIMACIÓN Y XML DE DATOS FÍSICOS

Se ha denominado "XML de animación" a los datos recolectados en Unity 3D o en Adobe Flash de la posición de cada objeto en cada frame. Entre los datos que se guardan en este fichero se encuentra el

tamaño de la escena en píxeles, la cantidad de objetos a manejar, el número de frames por segundo, o la posición de cada objeto para cada frame (x, y, ángulo).

Se ha denominado "XML de datos físicos" a aquel archivo XML que contiene las medidas reales del área de trabajo, posiciones del brazo robótico y cualquier otro dato relativo a la parte hardware del sistema. Este archivo contiende campos como los puntos de referencia físicos que definen la escena, el punto en el que el brazo robótico se encuentra fuera del campo de visión de la cámara, el punto sobre la escena, o el punto sobre la biblioteca de objetos.

La "biblioteca de objetos" es el lugar definido para colocar los objetos inicialmente. Se define por un punto, offsets de posición a partir de ese punto para cada objeto, así como la altura de cada objeto. Estos datos también se guardan en el XML de datos físicos, y son necesarios para que el robot pueda realizar el pick&place de cada uno de los objetos.

4 ESTRUCTURA INTERNA DEL SOFTWARE

El diseño del software se ha realizado siguiendo algunas de las directrices dadas por los patrones GRASP [5]. En la Figura 4, se muestra un diagrama de bloques que representa la asignación de responsabilidades de creación y la composición para lo que se han aplicado principalmente los patrones de diseño Creador y Experto en Información. De esta manera el objeto Form es responsable de crear AutoMovement (con los datos recibidos del usuario) e inicializar el Robot y la cámara. El módulo AutoMovement llama al módulo de extracción de datos del XML para obtener los movimientos a ejecutar, y para ello utiliza el módulo (XML_Handler) que es el experto en lectura de XML y por tanto se encarga de extraer esta información de los mismos. Una vez extraídos estos datos y convertidos a milímetros, son guardados en el objeto physicalProperties que se encarga de suministrarlos al componente Automovement, según sea necesario.

Automovement una vez tiene los datos y acceso a la Webcam y al robot inicializado, ejecuta la secuencia de movimiento con la toma de imágenes en cada frame. Tanto la toma de imágenes, como la ejecución de las acciones del robot se realizan en hilos de ejecución independientes, evitando así bloquear la interfaz de usuario.

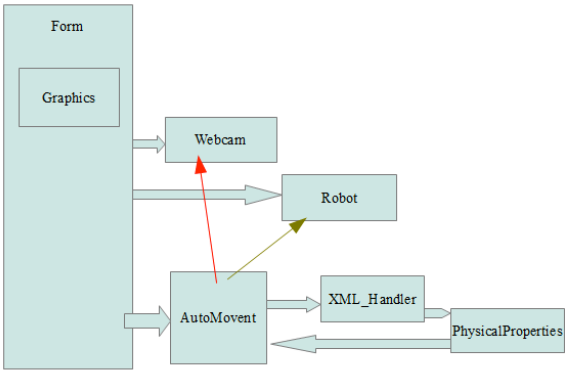

Figura 4: Estructura interna del Software

La casilla de verificación "Return all objects to Lib each frame" de la pestaña "XML Script", permite dos modos diferentes de funcionamiento del sistema. El primero de ellos, cuando la casilla se encuentra activa el robot mueve los objetos al almacén después de realizar la captura de imagen de cada frame, mientras que en el caso en que la casilla se encuentra inactiva, el robot desplaza los objetos entre imágenes sucesivas por el entorno, sin retornarlos al almacén.
De esta forma el primer caso, con la casilla "Return all objects to Lib each frame" activada, la secuencia de acciones del robot es la siguiente:

1. Coloca una a una cada pieza de la biblioteca de objetos en la escena.
2. Vuelve a la posición sobre la biblioteca de objetos. (Robot no visible en la escena)
3. Toma una captura de pantalla.
4. Devuelve todos los objetos a la biblioteca.
5. Se repite a partir del punto 1 hasta finalizar la secuencia de movimientos cargados.

Este proceso tiene la desventaja de ser lento al tener que mover uno a uno todos los objetos hasta su posición inicial en el almacén entre cada toma de imágenes.

En el segundo caso, el robot realiza las siguientes acciones:

1. Coloca una a una cada pieza de la biblioteca de objetos en la escena.
2. Vuelve a la posición sobre la biblioteca de objetos. (Robot no visible en la escena)
3. Toma una captura de la escena.
4. Mueve objeto a objeto desde su posición actual a la siguiente.
5. Se repite a partir del punto 2 hasta finalizar la secuencia de movimientos cargados.

Este modo tiene la ventaja de ser relativamente más rápida que el caso anterior, sin embargo requiere de un proceso de optimización para planificar en qué

orden deben moverse los objetos, teniendo en cuenta que no se desplace nunca un objeto sobre otro que aún está por moverse. La principal desventaja con respecto al caso anterior reside en que este problema no siempre tiene solución, por ejemplo el caso en que dos objetos intercambien sus posiciones en frames sucesivos, lo que puede llevar a obtener resultados no deseados en la animación.

Visualmente la aplicación tiene el aspecto mostrado en la Figura 5. A continuación se describe cada uno de los bloques funcionales resaltados en la Figura 5:

1. Selección de los ficheros de entrada. En este bloque se pueden cargar los archivos XML descritos en la sección 3.3.
2. Una vez seleccionados los ficheros, con este botón se importan los datos. Si la importación se ha realizado con éxito, los campos de "Parameters from files" estarán actualizados con el número de frames (Nr of frames), el número de objetos (Nr of objects), etc.
3. Con estos controles podemos enviar la posición en articulares al robot o bien sobre el área de la librería de objetos, o bien sobre la escena. Posteriormente se describirán estos conceptos.
4. La ruta en la que se guardan las imágenes.
5. Frame donde empezamos a grabar imágenes y número de frame actual.
6. Estas opciones permiten intercambiar los ejes X-Y (para realizar un pseudo-giro) y para devolver todos los objetos a la librería de objetos entre frame y frame.
7. Botones de marcha/paro.

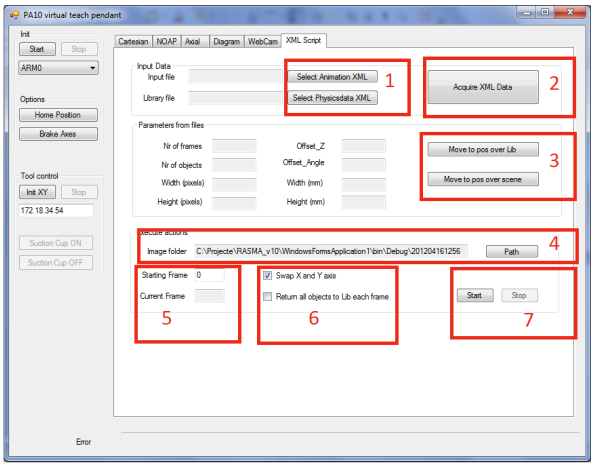

Figura 5: Interfaz de la aplicación

5 RESULTADOS

5.1 ARQUITECTURA DEL SISTEMA

La plataforma RASMA desarrollada cuenta como elemento encargado del posicionamiento principal con un brazo robótico Mitsubishi PA-10 de 7 grados de libertad con controladora ISA. En el extremo del robot se ha acoplado una herramienta ventosa accionada mediante un controlador neumático. La controladora ISA está conectada a un PC industrial donde se aloja el programa de la plataforma RASMA. La captura de los distintos fotogramas que compondrán la animación de Stop-motion se realiza mediante una webcam Logitech USB colocada cenital y perpendicularmente al área de trabajo. La Figura 6 muestra el brazo robótico y el espacio de trabajo donde se va a crear la escena de animación.

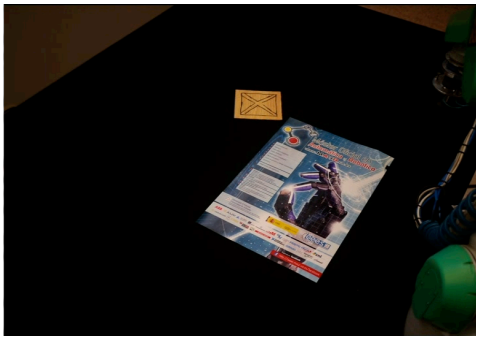

Figura 6: Composición de la escena mediante el brazo robótico

En cuanto a los elementos Software utilizados en la plataforma RASMA, cabe destacar el programa "PA10 virtual tech pendant", creado específicamente para esta aplicación y con funciones de control de la cámara web, lectura de ficheros XML y control del brazo robótico Mitsubishi PA-10. Este programa se ha creado con Visual Studio Express 2010. Para la generación de los XML de animación se han utilizado tanto Unity 3D, versión gratuita (http://unity3D.com/), y opcionalmente el Toolkit 2D (http://unikronsoftware.com/2dtoolkit/), como Adobe Flash Professional CS5. Para la captura de imágenes a través de la webcam se ha utilizado la librería EasyWebCam Library C# (http://easywebcam.codeplex.com/). Finalmente, se ha utilizado Luciole sobre Ubuntu para la composición del vídeo a partir de los fotogramas generados. Luciole permite resultados rápidos y con buena compresión (xvid) aunque las opciones son limitadas. Si se está utilizando otro software, la opción "onion skinning" proporciona una sensación de movimiento más realista.

5.2 EXPERIMENTO

A continuación se muestra la creación de una secuencia de ejemplo generada con Unity 3D según la Figura 2. Para calcular la trayectoria se especifica simplemente la posición inicial, una fuerza aplicada al objeto, el número de fotogramas a capturar (20) y el número de frames por segundo (6). El programa calcula la posición de la caja para cada fotograma, generando así un XML de animación, que junto con un XML de datos físicos de la escena, se utiliza por el software creado para el brazo robótico PA-10 que coloca la pieza en su posición automáticamente para cada fotograma y se aparta a la hora de tomar la captura de la imagen. La Figura 7 muestra la secuencia generada para este experimento.

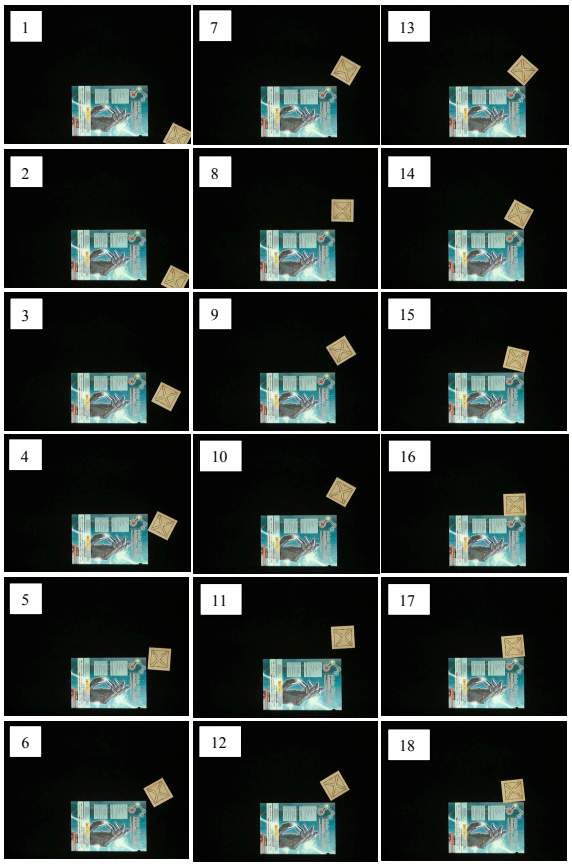

Figura 7: Secuencia de la animación Stop-motion generada con la aplicación propuesta

6 CONCLUSIONES

Con el método y el software descrito es posible realizar animaciones Stop-motion con asistencia robótica. Se demuestra que se pueden generar trayectorias y efectos más precisos que con el método tradicional, y con un robot posicionador rápido, la generación de los fotogramas sería también más rápida.

En las primeras pruebas de la plataforma RASMA descrita en el presente artículo se han obtenido animaciones generadas de forma desatendida con gran precisión. En esta primera aproximación a una plataforma de creación automática de animaciones Stop-motion se han detectado una serie de mejoras que se pretenden desarrollar en trabajos futuros. El uso de un robot paralelo permitiría una mayor frecuencia de trabajo. Otro trabajo que se plantea para mejorar la plataforma RASMA es el empleo de la visión para detectar los elementos situados en la escena y a partir de estos elementos realimentar de manera automática la información de posición y tamaño de las piezas para componer la animación en Unity o en Adobe Flash. Por último, la aplicación actual no permite la generación del vídeo a partir de los fotogramas capturados de la escena. En esta fase de la creación de la animación una posible mejora de la plataforma debería permitir componer el vídeo a partir de los frames sin tener que recurrir a una aplicación externa, así como el empleo de un método software para añadir efecto *motion blurring* a la secuencia de fotogramas en vez del *onion skinning* [6].

Agradecimientos

Este trabajo está financiado por el Ministerio de educación y ciencia (Proyecto DPI2011-22766).

Referencias

[1] Bonanni, L., Ishii, H., (2009) *Stop-motion prototyping for tangible interfaces*. TEI '09 Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, pp. 315-316.

[2] Raffle, H., (2004). *Topobo: A 3-D Constructive Assembly System with Kinetic Memory*. Master's Thesis, MIT.

[3] Arden, S. *Whizard Motion*. http://www.ecuad.ca/research/practice-basedresearch/graduate.

[4] Jiang, Y., Zheng, C., Lim, M., Saxena, A., (2011) *Learning to Place New Objects*. arXiv:1105.3107v2 [cs.RO].

[5] Larman, C., (2005) *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.)*. New Jersey: Prentice Hall.

[6] Brostow, G. J., Essa, I., (2001) *Image-based motion blur for stop motion animations*. Proceeding SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 561-566.