Modelado de controles tangibles para juegos con ToyVision

Javier Marco
Madeira-ITI
University of Madeira, Portugal
javier.marco@m-iti.org

Eva Cerezo, Sandra Baldassarri
Advanced Computer Graphics Group (GIGA)
Computer Science Department,
Engineering Research Institute of Aragon (I3A)
University of Zaragoza, Spain
{ecerezo, sandra}@unizar.es

ABSTRACT
El presente articulo presenta "*ToyVision*", un toolkit software para el prototipado rápido de interfaces tangibles en superficies horizontales interactivas o tabletops. La arquitectura de este toolkit incorpora una capa de abstracción de alto nivel, el nivel de dispositivo, que permite a los desarrolladores modelar en alto nivel los controles tangibles en juegos para tabletop, e independizar el desarrollo del juego de las plataformas hardware y software utilizadas. En el nivel de hardware *ToyVision* se basa en el entorno opensource Reactivision, al que se le han añadido nuevas funcionalidades. El toolkit comprende también un asistente gráfico que recoge del diseñador toda la información necesaria para el modelado de la pieza del juego en *ToyVision*. Para mostrar la potencia del toolkit propuesto se muestran al final del artículo dos juegos, con sus correspondientes controles, generados gracias a las características de *ToyVision*.

Author Keywords
Tabletop, toolkit, tangible, juegos, juguetes, arquitectura.

ACM Classification Keywords
H5.2. [Input devices and strategies]: User Interfaces.

General Terms
Design.

1. INTRODUCCIÓN
La introducción de los dispositivos tabletop (superficies interactivas) tanto a nivel académico [23] [25] [7] como comercial [21] [24] está abriendo las puertas a una nueva generación de videojuegos físicos y sociales, híbridos entre el juego de tablero tradicional y las nuevas posibilidades de aumentar computacionalmente el espacio de interacción a través de imagen y sonido. La mayor parte de estos juegos se basan en la interacción multitáctil [6] [2]. Al desarrollar juegos de tablero para este estilo de interacción, las piezas del juego son representadas virtualmente en la superficie activa y los usuarios las manipulan mediante gestos con sus dedos. Muchos de los dispositivos tabletop no sólo detectan las manipulaciones de los usuarios con las manos, sino que además son capaces de detectar e identificar objetos convencionales manipulados sobre la superficie. Gracias a esta posibilidad están apareciendo ejemplos de juegos basados en interacción tangible para dispositivos tabletop [11] [17]. Manteniendo en el entorno físico las piezas manipulables se amplifica el impacto emocional del juego de ordenador [14] [13] y se hace la tecnología más accesible a otros tipos de usuarios como son los niños pequeños [20], personas con discapacidad [18], y personas mayores [1].

Sin embargo, los diseñadores de juegos para estos dispositivos todavía tienen que enfrentarse a codificar en bajo nivel los algoritmos de detección de las manipulaciones del usuario y de identificación de las piezas físicas del juego, ralentizando el prototipado rápido de los juegos, e impidiendo su reutilización en dispositivos tabletop con hardware distinto.

En este artículo se presenta *ToyVision*, un toolkit para el desarrollo en alto nivel de prototipos de juegos tangibles para dispositivos tabletop. Gracias a él, el desarrollador puede modelar de forma gráfica las piezas físicas que se van a utilizar en el juego, abstrayéndose de la implementación de la detección de estas piezas por el dispositivo tabletop. Para llevar a la práctica esta propuesta se ha modificado el entorno opensource Reactivision [15] y se ha implementado un nuevo nivel de abstracción, el nivel de dispositivo, que ha sido desarrollado para el entorno de desarrollo Adobe Flash. No obstante, lo expuesto en este artículo puede ser fácilmente replicado en otros entornos para tabletops y entornos de desarrollo de aplicaciones para tabletop.

2. ESTADO DEL ARTE
Las ventajas de encapsular el desarrollo de interfaces de usuario en niveles de abstracción han sido ampliamente exploradas [9] y aprovechadas en varios toolkits para el desarrollo de GUIs bajo el paradigma WIMP.

El éxito que en los últimos años han tenido las aplicaciones multitáctiles también ha llevado a aplicar arquitecturas basadas en niveles de abstracción, dando lugar a diversos toolkits que permiten el desarrollo de aplicaciones con independencia del dispositivo tabletop y el entorno de desarrollo [28] [27] [19] e incluso abstrayendo al desarrollador del reconocimiento de los gestos realizados por los usuarios [12] [10].

Aplicar el modelo de interacción tangible en dispositivos tabletop requiere la detección y seguimiento de objetos convencionales sobre la superficie. La detección de objetos físicos en dispositivos de tipo tabletop es un campo reciente de investigación. La mayor parte de las interfaces tabletop hacen uso de técnicas de visión por

computador basadas en iluminación infrarroja para la detección de las yemas de los dedos como blobs blancos [26]. Con estas técnicas también es posible identificar y seguir objetos a partir de su forma. Sin embargo, una técnica mucho más robusta consiste en pegar a la base de los objetos un marcador impreso (fiducial). Muchos dispositivos tabletop ofrecen su propia colección de marcadores impresos para reconocer cualquier objeto. Microsoft Surface usa "DotCode" [4], Reactable usa "Reactivision amoebas" [15], que gracias a su carácter de código libre, se ha convertido en un estándar "de-facto" y está siendo soportado por otros toolkits como CCV [5]. Existen otras soluciones de fiduciales no asociadas a tabletop comerciales, como ARTOOLKIT [3], Trackmate [29], y Cybercode [16]. Todos ellos se basan en la misma idea: el fiducial se compone de áreas reflectivas y áreas no reflectivas a la luz, en una distribución topológica distinta en cada fiducial, lo que permite, a partir de los blobs de las áreas reflectivas, identificar las distintas distribuciones pertenecientes a los distintos objetos. El sistema puede por tanto, diferenciar objetos y dar su posición en la superficie del tabletop. Si el fiducial se diseña de forma asimétrica, puede también extraerse la orientación del objeto. Existen incluso técnicas para hacerlos invisibles al ojo humano pero visibles para la cámara IR [22]. El desarrollo de fiduciales está permitiendo que muchos toolkits multitáctiles también soporten ya el desarrollo de aplicaciones tangibles para tabletops.

La arquitectura de las aplicaciones toolkit para dispositivos tabletop ha sido descrita por Echtler and Klinker [8] a través de 4 niveles de menor a mayor grado de abstracción: nivel de hardware, nivel de calibración, nivel de interpretación de eventos y nivel de dispositivo (ver fig. 1). Todos los toolkits para el desarrollo de aplicaciones para tabletop ofrecen, al menos, el nivel de hardware, abstrayendo al desarrollador de acceder a los dispositivos de reconocimiento visual, y a los algoritmos de reconocimiento de blobs y de fiduciales. Conforme se van añadiendo niveles de abstracción, el desarrollador de aplicaciones para tabletop recibe del toolkit información de más alto nivel de las interacciones del usuario en la superficie. La separación entre el toolkit y el entorno de desarrollo de la aplicación para tabletop se está garantizando actualmente mediante la comunicación por medio de protocolos de red. En este sentido, el protocolo TUIO [15] se está convirtiendo en un estándar "de facto" para comunicar toolkits multitáctiles y tangibles con una amplia colección de entornos de desarrollo. Gracias a ello se adquiere la posibilidad de desarrollar aplicaciones con independencia del dispositivo tabletop. Sin embargo, el protocolo TUIO está diseñado para que el toolkit envíe eventos multitáctiles y tangibles (aparición, movimiento y desaparición de dedos y objetos) a la aplicación para el tabletop (ver fig. 1). Esto es así debido a que la mayoría de los toolkits existentes actualmente no implementan niveles de arquitectura más allá del nivel de interpretación de eventos. Poder acceder al nivel de abstracción de dispositivo le permitiría al desarrollador trabajar al nivel de los objetos que se encuentran sobre la mesa, y de sus estados en función de las manipulaciones de los usuarios.

La dificultad en diseñar un toolkit que implemente el nivel de abstracción de dispositivo, radica en la enorme amplitud de objetos del mundo físico que pueden ser usados en aplicaciones tabletop. Sin embargo, dado que el contexto en el que este trabajo quiere moverse es el del desarrollo de juegos de tablero, el espectro de objetos puede modelarse en unas pocas categorías, fácilmente gestionables en un nivel de dispositivo como se explica en el siguiente apartado.

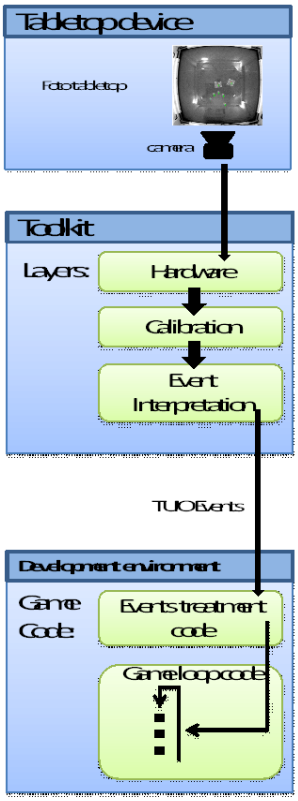

Figura 1. Arquitectura actual de los toolkit de desarrollo para dispositivos tabletop

3. TOYVISION: CATEGORÍAS DE OBJETOS TANGIBLES EN JUEGOS DE MESA

Cuando hablamos de juegos de mesa, no nos estamos limitando solo a juegos de tablero en los que varios jugadores se enfrentan manipulando piezas en un área dividida en áreas o casillas. Queremos cubrir el mayor número de actividades lúdicas que se juegan en una mesa utilizando objetos, incluyendo actividades plásticas como modelar con plastilina o pintar. Atendiendo a la función que los objetos cumplen en el juego, hemos distinguido las categorías que se exponen a continuación.

3.1 Simple Token

Es el objeto habitual de los juegos de tablero: las fichas (tokens) que cada jugador tiene y se van distribuyendo por el tablero en relación a determinadas reglas de juego. Son el elemento fundamental de juegos como las damas, el parchís, la oca o la ruleta. En general, los tokens son fichas pequeñas circulares idénticas entre sí, salvo por su color que suele usarse para representar al jugador al que pertenecen, o un valor económico, como en la ruleta. En general existen en el juego un gran número de tokens, pero de pocos tipos distintos (tokens blancos y negros en las damas, rojo, verde, azul y amarillo en el parchís….).

Para que el toolkit del dispositivo tabletop pueda reconocer tokens, es necesario identificar los blobs de las piezas colocadas directamente sobre la mesa. En caso de que el juego solo disponga de un tipo de piezas (todas exactamente idénticas entre sí), su reconocimiento visual puede hacerse directamente por la forma y

tamaño del blob capturado por el hardware visual (círculos de una determinada área). Sin embargo, si se requiere distintos tipos de ficha, la base de cada una se identifica por un fiducial reconocible por el toolkit. En el caso de *ToyVision*, cuyo nivel de hardware se basa en Reactivisión, la colección de fiduciales amoebas tiene una complejidad en su dibujo que hace que su reconocimiento sea inviable en piezas de menos de 4 centímetros de diámetro. Este es un tamaño excesivo para poder implementar juegos de tablero con un elevado número de piezas, como las damas o el parchís. Para poder utilizar fichas de tamaños más normales (1 cm de diámetro), se ampliado el algoritmo de reconocimiento de fiduciales de Reactivision para poder identificar fiduciales extremadamente sencillos basados un manchas negras, dentro de un área circular blanca (ver fig. 2). De esta forma, se pueden crear un número reducido de tipos de tokens distinguidos por el número de manchas dentro de su fiducial (en nuestras experiencias conseguimos reconocer cuatro grupos distintos de tokens, lo que permite implementar con *ToyVision* juegos como el parchís, con cuatro jugadores). Además, estos fiduciales pueden hacerse asimétricos, desplazando las manchas del centro, y con ello, el toolkit puede obtener la orientación del token, en caso de que el juego pueda requerirla.

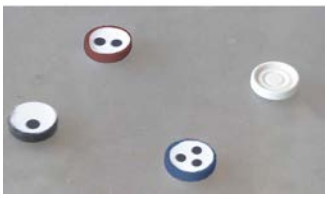

Figura 2. Diferentes fichas de juego con los nuevos fiduciales soportados por ToyVision

3.2 Named Token

Otros juegos también se componen de fichas, pero en este caso cada una tiene un aspecto distinto del resto, ya que cumple una función en el juego específica y única. Un claro ejemplo sería el ajedrez: la torre tiene unos movimientos distintos del alfil… Aunque puede haber varias instancias de la misma pieza (en el ajedrez hay dos torres, ocho peones…) siempre cada una está identificada, no sólo por el jugador al que pertenece, sino por la función que cumple. Otros juegos de mesa tienen una única instancia y además no identifican al jugador, como por ejemplo, los juegos de cartas: cada naipe es único en la baraja y tiene un nombre especifico: as de diamantes, tres de picas…

Para que el toolkit identifique cada pieza, la base de cada una lleva incorporado un fiducial único para cada tipo de pieza. En el caso de *ToyVision*, se ha aprovechado la colección de fiduciales amoebas que ya ofrece el toolkit Reactivisión (ver fig. 3). Ello afecta a la restricción de tamaño de piezas antes mencionada, por lo que se limita el número de piezas que podrán colocarse a la vez en la superficie.

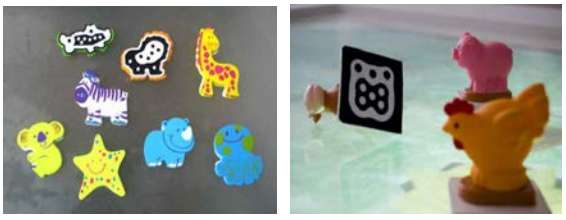

Figura 3. Diferentes juguetes identificados mediante fiduciales

3.3 Constraint Tokens

Otras piezas en juegos no son simples, sino que se componen de varias piezas que son manipulables dentro de los confines de la pieza principal. En la Figura 4 se muestra un control de este tipo. Podría ser usado para representar a cada jugador en el tablero (estilo pieza del Trivial Pursuit™) o como control de almacenamiento. Se compone de otras piezas más pequeñas (*tokens*) que se van colocando (o quitando) dentro de la pieza. En otros juegos, la manipulación puede ser más compleja; en una ruleta, en la que se hace girar una bolita alrededor de un plato circular, el *token* (la bolita) estaría restringido por el objeto mayor (ruleta).

Para que el toolkit identifique este tipo de piezas, se combina la detección de simple *tokens* y de *named tokens*. En *ToyVision* el objeto *constraint token* es identificado mediante un fiducial amoeba, mientras que los *simple tokens* se identifican por sus áreas blancas circulares (ver fig. 4).

Figura 4. Izquierda: Ejemplo de *constraint token* Derecha: base del juguete, fiducial para identificación y *simple tokens* con base blanca

3.4 Deformable Token

En otros juegos, los objetos no mantienen una forma constante, sino que ésta cambia al ser manipulados por los jugadores. Son generalmente materiales que se usan en juegos creativos, como la plastilina para modelar, cartulinas para recortar, o las pinturas.

Para que el toolkit reconozca la colocación de este tipo de elementos sobre el tabletop no se puede recurrir al uso de fiduciales, ya que es físicamente imposible añadir un marcador a algo que no tiene forma definida. En *ToyVision* la detección se basa en el reconocimiento visual de la forma del objeto. Para ello se ha ampliado en Reactivisión el algoritmo de reconocimiento visual de blobs para extraer información de su forma: área, perímetro, ángulos de inercia y una descripción vectorial del contorno del blob. Con esta información se elabora una lista de blobs presentes en cada momento en la superficie del tabletop, y se codifica en lenguaje XML, de forma que describe con detalle la geometría de todos los objetos no identificados con fiducial colocados en la mesa (ver fig. 5). Está lista se envía, con una periodicidad de tiempo configurable en el toolkit, a la aplicación que ejecuta el juego a través de un socket TCP-IP. Así, la aplicación del juego puede obtener la forma de los tokens deformables colocados durante el juego.

4. TOYVISION: ASISTENTE GRÁFICO Y NIVEL DE DISPOSITIVO

Para la implementación de *ToyVision*, con el nivel de dispositivo que lo distingue, se ha seguido un enfoque análogo a las herramientas existentes para la creación de interfaces GUI en aplicaciones basadas en WIMP, las cuales asisten al desarrollador

con herramientas gráficas para la creación de los controles de la aplicación. Posteriormente, el entorno de desarrollo permite el acceso a los controles creados a través de un conjunto de clases instanciadas para cada control incluido en el interface. *ToyVision* implementa también estas dos herramientas para el desarrollador (ver fig. 6):

1. Un **asistente gráfico** para la modelización de cada objeto tangible del juego, del cual saldrá la información necesaria para que los niveles inferiores del toolkit sepan qué tipo de elementos deben detectar, y cómo informar de ellos al nivel de dispositivo durante la ejecución del juego. Esto se lleva a cabo mediante ficheros de configuración que se cargan en la ejecución de la aplicación. Este asistente ha sido implementado en el entorno de desarrollo Adobe Flash tal como se detalla en la siguiente sección.
2. Un conjunto de **paquetes** y **clases** de programación Action Script 3 para Adobe Flash que permite al desarrollador el acceso a la información procesada por el nivel de dispositivo del toolkit. Esta información aparece encapsulada mediante una lista de objetos presentes en la superficie del tabletop y es continuamente actualizada según las manipulaciones de éstos. Esta librería de clases forma propiamente el nivel de dispositivo de la arquitectura propuesta, y será descrita con más detalle más adelante.

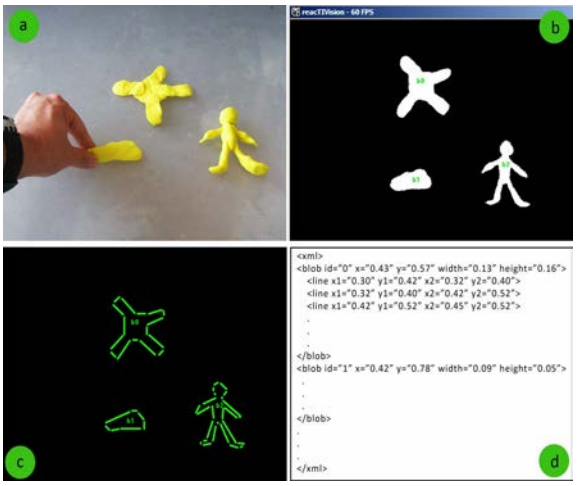

Figura 5. a: figuras de plastilina sobre el tabletop. b: imagen de los blobs detectados. c: detección de contornos de cada blob. d: codificación XML de cada blob en *ToyVision*

Como se puede ver en la Figura 6, en *ToyVision* se ha optado por implementar el nivel de dispositivo dentro del entorno de desarrollo de los juegos para tabletop, y no dentro del propio toolkit con el resto de niveles de la arquitectura. De esta forma se permite la reutilización del nivel de dispositivo por otros toolkits presentes o futuros de dispositivos tabletop, siempre y cuando se mantengan los protocolos de comunicación por red entre el nivel de dispositivo y el resto de niveles de arquitectura del toolkit. En ToyVision el protocolo de comunicación TUIO para la transmisión de eventos al nivel de dispositivo se ha complementado con una transmisión XMLSocket para enviar información ampliada sobre los *deformable tokens*, con objeto de mantener el protocolo TUIO dentro del estándar. En la futura actualización del protocolo TUIO 2.0 está previsto que soporte el envío de información de blobs y sus descriptores geométricos [15], lo que permitirá integrar el envío de esta información dentro del protocolo TUIO y prescindir de la conexión XMLSocket complementaria.

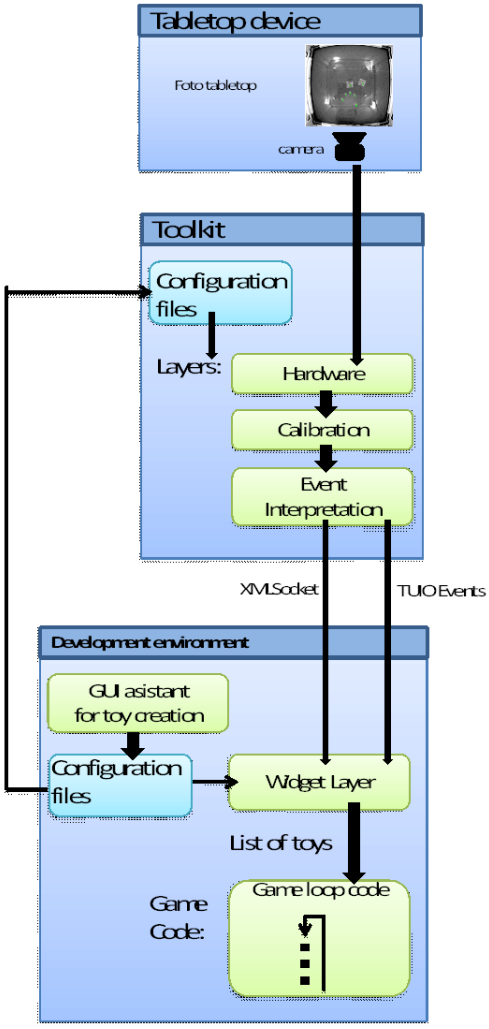

Figura 6. Arquitectura del toolkit *ToyVision*

A continuación se detallan el asistente gráfico y el nivel de dispositivo de *ToyVision*.

4.1 Asistente gráfico

Este módulo permite la modelización de forma gráfica de cada objeto tangible implicado en el juego a desarrollar. La aplicación funciona a modo de asistente, obteniendo paso a paso información sobre el juguete que se está modelando.

En primer lugar, el diseñador crea un nuevo control tangible eligiendo la categoría a la que pertenece: *simple token*, *named token*, *constraint token* o *deformable token* (ver fig. 7).

Una vez elegida la categoría el asistente solicita el nombre del juguete, el cual será usado para identificarlo en el nivel de dispositivo durante el desarrollo del juego. Tras ello se pasa a la interfaz de modelado del juguete: el diseñador recibe la imagen captada por la cámara del tabletop para que la use como referencia

al definir los elementos que el asistente le solicita, los cuales dependen de la categoría seleccionada.

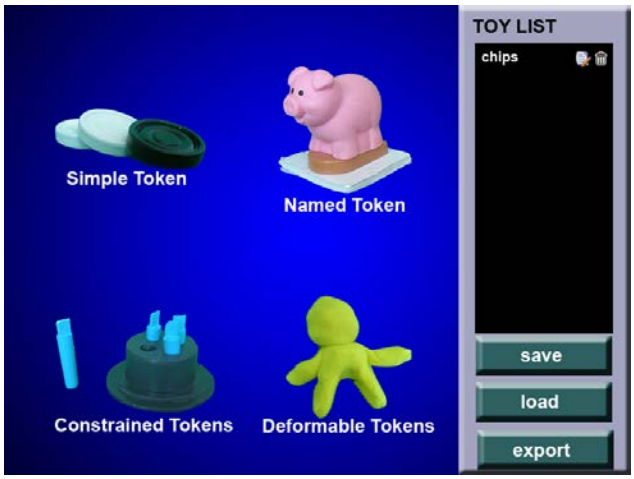

Figura 7. Menú principal del asistente gráfico de ToyVision

Simple tokens

- A partir de la imagen de la base de una ficha de juego el diseñador define gráficamente el tamaño del *token* y un rango de tolerancia. De esta forma, si se colocan objetos fuera de la tolerancia de tamaño definida, no serán identificados como piezas del juego y serán obviados.
- El diseñador define el número de *tokens* distintos que pueden ser detectados (en nuestra implementación hasta 4), y el número de piezas de cada tipo que hay en el juego.

Named Tokens

- El diseñador define, usando como referencia la imagen de la base del objeto, el área útil donde se podrá añadir físicamente el fiducial (ver fig. 8).
- Se define también el número de copias de cada objeto *named token* que puede haber en juego

Figura 8. Izquierda: juguete *named token*. Derecha: asistente gráfico de creación del *named token*, para definir el área del fiducial.

Constraint Tokens

- Primero, al igual que en el caso de un *named token*, se define el área útil de la base del objeto para la colocación del fiducial.
- Después se definen las áreas de manipulación de los *simple tokens* restringidos, definiendo previamente el tipo de dichas áreas: asociativas o manipulativas.

  Áreas asociativas son áreas dentro de las que pueden aparecen uno o más *simple tokens*. El diseñador dibuja en este caso una o varias áreas dentro de las cuales se podrá detectar la aparición de *simple tokens* (ver fig. 9).

Figura 9. Izquierda: juguete compuesto por cuatro *simple tokens* y áreas asociativas. Derecha: asistente gráfico de creación del juguete con herramientas para definir las áreas de aparición de los *simple tokens*

  Áreas manipulativas: son áreas dentro de las cuales un simple token puede ser desplazado o rotado. El diseñador define un área rectangular dentro de la cual se restringe el movimiento del token.

- Finalmente se detalla el número de copias de este objeto que pueden aparecer en el juego

Deformable tokens

En este caso el asistente no solicita al diseñador la definición gráfica el objeto ya que este puede tomar cualquier forma. Lo que se define es la manera en la que la información procesada de los blobs (tal como se ha descrito en el apartado anterior) es enviada a través de la conexión XMLSocket a la aplicación que ejecuta el juego en desarrollo:

- Por intervalo de tiempo: el diseñador define un intervalo de tiempo en segundos, en el cual se envía periódicamente la información XML de los blobs detectados en la superficie de la mesa.
- Por aparición de un objeto *named token*. Se asocia el envío del XML a la aparición en la superficie de la mesa de un objeto *named token* previamente definido en el interface.

Una vez se han definido todos los objetos tangibles del juego, el modulo gráfico exporta los ficheros de configuración XML necesarios para el toolkit del tabletop y el nivel de dispositivo con los parámetros definidos para cada objeto. Se genera también un diccionario de fiduciales a identificar en el juego. Además, se genera un fichero PDF con los fiduciales para los *simple, named y constraint tokens* definidos previamente, el cual, puede ser impreso y recortado para pegar cada fiducial en su correspondiente objeto, con lo cual quedaría listo para ser usado en el juego (ver fig. 10).

Figura 10. a: Hoja impresa de fiduciales para un juego. b: fiducial recortado y aplicación de pegamento. c: pegado del fiducial al juguete

4.2 Nivel de dispositivo: implementación del juego

El nivel de abstracción de dispositivo ofrece al desarrollador del juego las herramientas de programación necesarias para acceder a información de alto nivel sobre los controles tangibles del juego, y las manipulaciones que los usuarios realizan con ellos. En *ToyVision* el nivel de dispositivo se ha implementado como un paquete de clases Action Script 3 para el entorno de desarrollo Adobe Flash. La clase principal es ToyList, la cual, al ser instanciada al comienzo del juego, carga los ficheros de configuración creados con el asistente gráfico previamente descrito. Una vez instanciada, ofrece al programador una lista con los controles tangibles presentes en la superficie del tabletop, y su estado. Los controles de la lista están identificados unívocamente por el nombre identificador que se les otorgó al crearlos en el asistente gráfico. Por tanto, un bucle normal de juego, consiste en recorrer los objetos de la lista, y tomar las acciones adecuadas en el juego para cada uno de ellos (ver fig. 11).

```
public function Game() {
  //instantiate List of Toys
  gameToys= new ToyList('path to
                    configuration XML file');
  While (true) {//game loop}
}
public function TabletopEvent(toy, eventType)
{
 switch (toy.name) {
    case 'name1':
      if (eventType="add") {//toy placed}
      if (eventType="removed")
                           {//toy removed}
      if (eventType="updated")
                        {//toy moved/rotated}
      If (eventType="constraint")
          {//toy.updatedConstraint
           // is the area that triggered
           //the toy.constraintEvent
           Switch (toy.updatedConstraint) {
            Case 't1':
             if toy.constraintEvent="add"
                  {//a simple token has been
                   //added into this area
                  }
             if toy.constraintEvent="removed"
                  {//a simple token has been
                   //removed from this area
                  }
             if toy.constraintEvent="updated"
                  {//a simple token has been
                   //moved in this area
                  }
            Break;
           Case 't2';
           …
      break;
    case 'name2':...
      ...   } }}
```

Figura 11. Nivel de dispositivo: esquema del código AS3 de un juego desarrollado con *ToyVision*

5. JUEGOS IMPLEMENTADOS

A continuación, para dar una idea de la potencia creativa de *ToyVision* a la hora de prototipar juegos tangibles para dispositivos tabletop se muestran algunos ejemplos de juegos creados gracias al nivel de dispositivo de *ToyVision*.

5.1 Juego de música (Secuenciador)

El secuenciador es un juego de música en el que los niños pueden crear ritmos de batería (y bajo) usando las clásicas fichas circulares del juego de las damas. La partitura utilizada para crear ritmos musicales se puede entender como una distribución de notas musicales (las fichas) sobre una superficie bidimensional (la mesa), en la que el eje horizontal representa el tiempo, y el horizontal los diferentes instrumentos de la batería, representados con diferentes colores (ver fig. 12). La partitura se reproduce de izquierda a derecha.

Figura 12: Izqda: Fichas usadas en el juego de música "Secuenciador". Dcha: Fichas colocadas sobre la partitura

Controles tangibles

- Fichas: son interpretadas por el toolkit como *tokens***.** La partitura mostrada en la mesa consiste en una rejilla de 8 instrumentos por 16 tiempos, por lo que sería posible incluir hasta 128 fichas para llenar toda la partitura de sonidos. Esto sería imposible usando fiduciales estándar de Reactivision, ya que, dado el tamaño mínimo que deberían tener las fichas, no cabrían todas en la mesa. Gracias a los nuevos fiduciales diseñados, estos pudieron ser añadidos sin problemas a fichas normales de juego de damas, cuyas dimensiones permiten llenar toda la partitura. Los nuevos fiduciales (ver fig. 12) permiten, además, que Reactivision distinga entre las fichas blancas y negras. En el juego dicha distinción se utiliza para variar el tono del instrumento sobre el que está colocada la ficha. En el caso de los instrumentos de percusión (platos, bombo…) las fichas negras tienen un sonido más agudo y corto que las blancas. En el caso de los instrumentos de cuerda (bajo), se ha utilizado un diseño de fiducial orientable, de forma que girando la ficha sobre la mesa, se varía el tono a lo largo de la escala musical, pudiendo componer una melodía de hasta 16 notas.
- Regulador: En el juego de la música, un regulador permite variar la velocidad a la que se reproduce el ritmo compuesto. Girando en una u otra dirección la manivela amarilla (ver fig. 13), la música se reproduce más rápida o más lenta. El juguete es un *constraint token*, en el que un *simple token* manipulativo puede girar a lo largo del perímetro del objeto. A partir del ángulo entre el fiducial central y el simple token se obtiene el valor de velocidad de reproducción de la música.

Figura 13: Regulador del ritmo en el juego de música 'Secuenciador'

- Memoria: En el juego de la música, la memoria (ver figs. 4 y 14) permite guardar ritmos compuestos en la mesa. Al colocar una ficha azul en alguno de los cuatro huecos disponibles, la configuración actual de fichas en la mesa, queda automáticamente asociada con dicho hueco. De esta forma, cada vez que se coloca una ficha azul en un hueco que tiene un ritmo asociado, este se reproduce automáticamente sin necesidad de colocar las fichas en la mesa. De esta forma pueden ser almacenados y mezclados entre sí hasta cuatro ritmos. La memoria es un *constraint token*, compuesto de cuatro áreas asociativas en las que se pueden introducir *simple tokens* (fichas azules).

Figura 14: Uso de la memoria para almacenar ritmos en el juego de música 'Secuenciador'

5.2 Juego de pintar

Se trata de una aplicación tangible para dibujar virtualmente sobre el tabletop (ver fig. 15). Se pueden usar diversos tipos de objetos para componer los dibujos.

Controles tangibles

- Pinceles: se usan pinceles convencionales para pintar sobre la mesa. El color se elige pasando el pincel sobre una paleta virtual. El grosor del trazo dependerá del grosor del pincel: el pincel es interpretado por el toolkit como un deformable token, y por tanto, el grosor del pincel es conocido a través del área del blob que genera el pincel apoyado sobre la superficie de la mesa
- Papeles: Los niños pueden pintar con rotuladores sobre folios marcados con un fiducial (named tokens). Después, colocando boca abajo el papel sobre la mesa, el dibujo queda registrado de forma virtual (el dibujo es interpretado como un deformable token por el toolkit, y la aparición del fiducial del papel, dispara el evento para enviar el dibujo a través del XMLSocket). Los dibujos quedan guardados y representados por iconos en la barra superior de la aplicación de dibujo.
- Tampones: Son una colección de named tokens, que se usan para "pegar" los dibujos virtuales capturados de los papeles. Los usuarios ponen el tampón encima del icono de dibujo de la barra superior, para que este quede "impregnado" con el dibujo. Después pueden ir pegando el dibujo en el lienzo virtual simplemente golpeando el tampón sobre la superficie de la mesa.

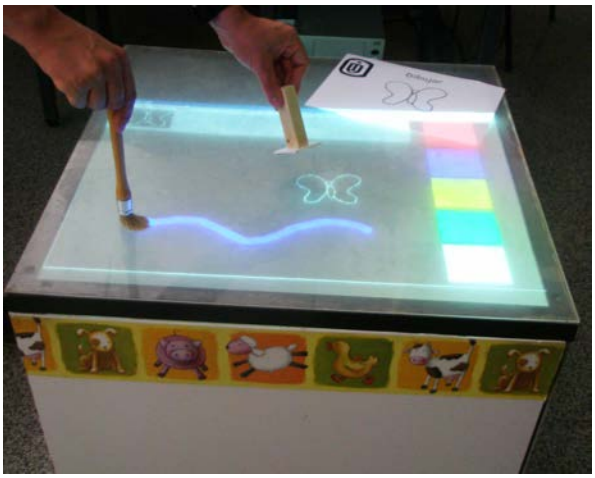

Figura 15: Controles tangibles del juego de pintar

6. CONCLUSIONES

El toolkit *ToyVision* proporciona a los desarrolladores y diseñadores de juegos para tabletops herramientas que les permiten el prototipado rápido de juegos y juguetes.

Una característica diferenciadora de *ToyVision* es la gran variedad de controles que permite considerar, lo que abre la puerta a nuevas posibilidades de interacción tangible en los dispositivos tipo tabletop. Ello ha sido posible mediante el añadido de nuevas funcionalidades a Reactivision, la base del nivel de hardware del toolkit. Un asistente gráfico le facilita al diseñador el modelado de cada pieza del juego y genera automáticamente su especificación en XML con toda la información necesaria para facilitar en el nivel de dispositivo la implementación del juego.

Y es que la diferencia fundamental entre *ToyVision* y otros toolkits similares es la implementación de un nivel de dispositivo que permite a los diseñadores afrontar el desarrollo de las aplicaciones o juegos con un mayor nivel de abstracción. El nivel de dispositivo proporciona acceso a una serie de clases AS3 que informan del status de cada una de las piezas del juego a lo largo de la partida.

El software ToyVision y su código fuente están disponibles para descarga bajo licencia GPLv3 en la siguiente URL: http://webdiis.unizar.es/~jmarco/?page_id=297&lang=es

Como trabajo futuro se plantea la expansión de *ToyVision* a otros entornos desarrollo y a otros tipos de juegos y aplicaciones tangibles.

7. AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el Gobierno Español a través del DGICYT contrato TIN2011-24660.

8. REFERENCIAS

1. Al Mahmud, A., Mubin, O., Shahid, S. and Martens, J.B. Designing and evaluating the tabletop game experience for senior citizens. 5th Nordic conference on Human-computer interaction (NordiCHI '08). ACM, pp.403-406.
2. Antle, A.N., Bevans, A., Tanenbaum, J., Seaborn, K., and Wang, S. 2010. Futura: design for collaborative learning and game play on a multi-touch digital tabletop. Fifth international conference on Tangible, embedded, and embodied interaction (TEI '11). ACM, pp. 93-100.
3. ARToolkit web: http://www.hitl.washington.edu/artoolkit/
4. Bollhoefer, K.W., Meyer, K., and Witzsche, R. Microsoft surface und das Natural User Interface (NUI). Technical report, Pixelpark, Feb. 2009.
5. CCV web: http://ccv nuigroup.com/
6. Cooper, N., Keatley, A., Dahlquist, M., Mann, S., Slay, H., Zucco, J., Smith, R., and Thomas, B. H. 2004. Augmented Reality Chinese Checkers. In Proceedings of the 2004 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology (2005). ACE '04, vol. 74. 117-126.
7. Dietz, P. and Leigh, D. DiamondTouch: a multi-user touch technology. In UIST ’01: Proceedings of the 14th annual ACM symposium on User interface software and technology, pages 219–226. ACM, 2001.
8. Echtler, F., Klinker G. A multitouch software architecture. In Proc of NordiCHI '08. 2008. pp. 463- 466.
9. Greenberg, S. Enhancing creativity with groupware toolkits. Groupware: Design, Implementation, and Use. Springer. 2003. pp. 1-9.
10. Hansen, T.E., Hourcade, J.P., Virbel, M., Patali, S., and Serra, T. PyMT: a post-WIMP multi-touch user interface toolkit. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09). ACM, New York, NY, USA, 17-24.
11. Heijboer, M., and van den Hoven, E. Keeping up appearances: interpretation of tangible artifact design. In Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges (NordiCHI '08). ACM, New York, NY, USA, 162-171.
12. Heng, X., Lao, S., Lee, H., and Smeaton, A. A touch interaction model for tabletops and PDAs. In Proc. PPD ’08, 2008.
13. Hinske, S. and Langheinrich, M. 2009. W41K: digitally augmenting traditional game environments. In Proceedings of the 3rd international Conference on Tangible and Embedded interaction (2009). TEI '09, 99- 106.
14. Iwata, T., Yamabe, T., Polojrvi, M., and Nakajima, T. Traditional games meet ICT: a case study on go game augmentation. In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction (TEI '10). ACM, New York, NY, USA, 237-240.
15. Kaltenbrunner, M. reacTIVision and TUIO: a tangible tabletop toolkit. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09). ACM, New York, NY, USA, 9-16.
16. Koike, H., Sato, Y., Kobayashi, Y., Tobita, H., and Kobayashi, M. Interactive Textbook and Interactive Venn Diagram: Natural and Intuitive Interface on Augmented Desk System. In Proceedings of Human Factors in Computing Systems (CHI’2000), pages 121–128. ACM.2000
17. Leitner, J., Haller, M., Yun, K., Woo, W., Sugimoto, M., Inami, M., Cheok, A. D., and Been-Lirn, H. D. 2010. Physical interfaces for tabletop games. Comput. Entertain. 7, 4, Article 61 (January 2010), 21 pages.
18. Li, Y., Fontijn, W., and Markopoulos, P. 2008. A Tangible Tabletop Game Supporting Therapy of Children with Cerebral Palsy. 2nd International Conference on Fun and Games, Springer-Verlag, Berlin, Heidelberg, 182-193.
19. Lin, H.H., and Chang, T.W. A camera-based multi-touch interface builder for designers. In Human-Computer Interaction. HCI Applications and Services, 2007.
20. Marco, J, Cerezo, E., Baldassarri, S. Mazzonne, E. Read, J.Bringing Tabletop Technologies to Kindergarten Children. 23rd BCS Conference on Human computer Interaction. Cambridge University. 1-4 september 2009. p.103-111. ISBN:978-1-60558-395-2.
21. Microsoft surface: http://www microsoft.com/surface/
22. Park, H., and Park, J.I. Invisible marker tracking for AR. In Proc. of Third IEEE and ACM Intl. Symposium on Mixed and Augmented Reality (ISMAR 2004). IEEE/CS, 2004.
23. Patten, J., Ishii, H., Hines, J. and Pangaro, G. Sensetable: a wireless object tracking platform for tangible user interfaces. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'01). ACM, 253-260.
24. Reactable web: http://www reactable.com
25. Rekimoto, J. and Saito, M. Augmented Surfaces: a spatially continuous work space for hybrid computing environments. In Proceedings of the ACM Conference on Human Factors in Computing System (CHI’99), pages 378–385. ACM, 1999.
26. Schöning, J., Hook, J., Motamedi, N., Olivier, P., Echtler, F., Brandl, P., Muller, L., Daiber, F., Hilliges, O., Löchtefeld, M., Roth, T., Schmidt, D. and von Zadow, U. Building Interactive Multi-touch Surfaces. JGT:Journal of Graphics Tools. 2009.
27. Shen, C., Vernier, F., Forlines, C., and Ringel, M. DiamondSpin: an extensible toolkit for around-the-table interaction. In Proc. CHI ’04, pages 167–174, 2004.
28. Touchlib. http://www nuigroup.com/touchlib/.
29. TrackMate web: http://trackmate.sourceforge net/