Componentes necesarios para implementar un Escritorio Estereoscópico

Gonzalo J. HERNANDEZ
Departamento de Sistemas, Universidad de Nariño
Pasto, Nariño, Colombia

y

Silvio R. TIMARAN
Departamento de Sistemas, Universidad de Nariño
Pasto, Nariño, Colombia

RESUMEN

El grupo de investigación GRIAS de la Universidad de Nariño inició un proyecto con el ánimo de analizar el comportamiento de un usuario frente a un computador utilizando un escritorio proyectado en 3 Dimensiones usando estereoscopía, para llevar a cabo este estudio fue necesario iniciar con la creación de un prototipo de escritorio estereoscópico el cual fue llamado *S-Desktop*.

El estudio realizado ha detectado que para la implementación de *S-Desktop* se requieren cuatro componentes, los cuales debieron ser puestos a prueba para analizar si era pertinente usarlos.

Este documento es el resultado de las pruebas realizadas a estos cuatro componentes, lo cual determinó que sí es posible utilizarlos para implementar *S-Desktop*.

Palabras Claves: Escritorio, Estereoscopía, 3D, GUI, HCI.

1. INTRODUCCIÓN

La mayoría de sistemas operativos disponen de una capa que permite al usuario final interactuar con el computador por medio de objetos gráficos, de esta forma el usuario no tiene que aprender comandos textuales para realizar tareas simples como: revisar el contenido de un dispositivo de almacenamiento, o ejecutar un programa previamente instalado, entre otras. Esta capa denominada "escritorio" utiliza principalmente la potencia de una tarjeta graficadora para producir íconos, figuras geométricas y demás símbolos sobre una pantalla que representan acciones posibles para el usuario final, el cual puede ser manipulado por medio del teclado y del mouse [1].

Con la llegada de nuevas tecnologías de video, las pantallas ahora también permiten visualizar video en forma estereoscópica gracias a la emisión de dos proyección simultaneas pero en diferentes frecuencias que son filtradas por gafas especiales y permiten entregar a los ojos del usuario final el mismo video pero desde una perspectiva diferente, logrando crear en el cerebro una imagen tridimensional.

A pesar de tener a disposición varios opciones de escritorios, tal como se plantea en la sección 2.1, no existe uno que sea proyectado en tres dimensiones utilizando técnicas estereoscópicas.

En el grupo de investigación GRIAS del Departamento de Sistemas de la Universidad de Nariño, esta desarrollando un proyecto de investigación que pretende analizar el comportamiento de un usuario frente a un escritorio estereoscópico, en el cual sea posible ver los objetos tales como ventanas, iconos, menús, etc, con diferentes profundidades. Para tal fin, el proyecto de investigación requiere crear un prototipo de escritorio estereoscópico.

En la sección 2 se detalla cada uno de los cuatro componentes requeridos para la implementación del escritorio estereoscópico. Finalmente, en la sección 3 se presentan las conclusiones y trabajos futuros.

2. COMPONENTES DE UN ESCRITORIO ESTEREOSCÓPICO.

Para implementar el prototipo de escritorio estereoscópico, se han estudiado cuatro diferentes elementos que deberán ser puestos en conjunto, Un servidor gráfico, una librería de objetos, un sistema de proyección de video estereoscópico y una librería de calculo para dibujar objetos en tres dimensiones. A continuación se describe cada uno de ellos.

2.1 Usando X Window System

El servidor gráfico X11, es un sistema de red transparente que permite que muchas aplicaciones puedan funcionar al mismo tiempo por medio de ventanas, generando gráficos y textos. Su característica permite que las aplicaciones puedan ser ejecutadas en la misma máquina o en otros computadores conectados a la misma red [10].

El servidor gráfico X11 es utilizado por los sistemas operativos de la familia Unix para desplegar un ambiente gráfico, el cual dispone de las funciones adecuadas para trabajar con la pantalla, el teclado y el mouse de un computador.

La implementación del sistema gráfico X11 es abierta gracias a la librería Xlib escrita en C++ [8], y hace posible crear nuevos ambientes que cumplan funciones como las que disponen Gnome, KDE, XFCE, OpenLook, entre otros [1].

En esta primera fase se buscó verificar el procedimiento necesario para iniciar modo gráfico utilizando X Windows System sin contar un escritorio previamente cargado.

En el estudio realizado se crearon diferentes aplicaciones gráficas utilizando la Xlib, lo cual determinó que sí es posible crear un entorno utilizando como base las funciones primitivas del servidor gráfico x11.

Utilizando la distribución Arch Linux, se hicieron también los ejercicios pertinentes para crear, compilar y poner en marca una sencilla aplicación gráfica que funcionara sin depender de otro escritorio. Para tal fin, en el computador linux se instalaron los paquetes xorg-server y xorg-xinit adicionales a los necesarios para compilar c++.

La figura 1 presenta el script creado en el "home" del usuario con el cual fue realizada la prueba. El archivo debe tener el nombre ".xinitrc" debido a que es el nombre esperado por el Sistema X para iniciar el sistema gráfico.

Figura 1. Script de .xinitrc

```
#!/bin/sh
./test-x11
```

En el archivo ".xinitrc", se debe listar las aplicaciones gráficas que serán cargadas por el Sistema X cuando se ejecute el comando "startx". En este caso se ha invocado a la aplicación "test-x11" cuyo código puede detallarse en la figura 2, el ejemplo utilizado fue tomado y modificado de [16].

Figura 2. test-x11.cpp

```
#include <X11/Xlib.h>
#include <stdio.h>
#include <stdlib.h>
int main (void) {
   Display* display;
   Window window;
   XEvent event;
   int s;
   display = XOpenDisplay( NULL );
   if (display == NULL) {
      exit(1);
   }
   s = DefaultScreen (display);
   window = XcreateSimpleWindow(
      display,
      RootWindow(display,s),0,0,
      XdisplayWidth(display,s),
      XdisplayHeight(display,s),1,
      WhitePixel(display,s),
      WhitePixel( display , s ));
   XselectInput(display,window,
      ExposureMask|KeyPressMask );
   XMapWindow(display,window);
   for (;;) {
      XNextEvent (display, &event);
      if (event.type == Expose) {
         XdrawRectangle(display,
         window,
         XdefaultGC(display,s),
         50,50,300,200);
         XdrawString(display,
            window,
            XdefaultGC(display,s),
            55,65,"Window",11);
         XdrawLine(display,
```

```
            window,
            XdefaultGC(display,s),
            50,70,350,70);
      }
      if (event.type == KeyPress) {
         break ;
      }
   }
   XCloseDisplay (display);
   return 0;
}
```

La ejecución de la prueba puede verse en la figura 3 cuando es ejecutado el comando startx el cual inicia el Systema X y dibuja la simulación de una ventana.

Figura 3. Ejecución del comando startx

2.2 Haciendo un Branching al proyecto SimpleGUI

Una vez determinado el modo en el cual el Sistema X permitirá la ejecución del nuevo escritorio, es necesario disponer de una librería que permita trabajar los objetos básicos de un escritorio, tales como: Ventanas, Botones, Iconos, Menús, Cajas de Texto, Etiquetas, Cajas de listados, entre otros.

El estudio ahora requirió probar un software ya creado que permitiera trabajar con objetos visuales que no requieran de ninguna plataforma mas que la proporcionada por *X Window System*.

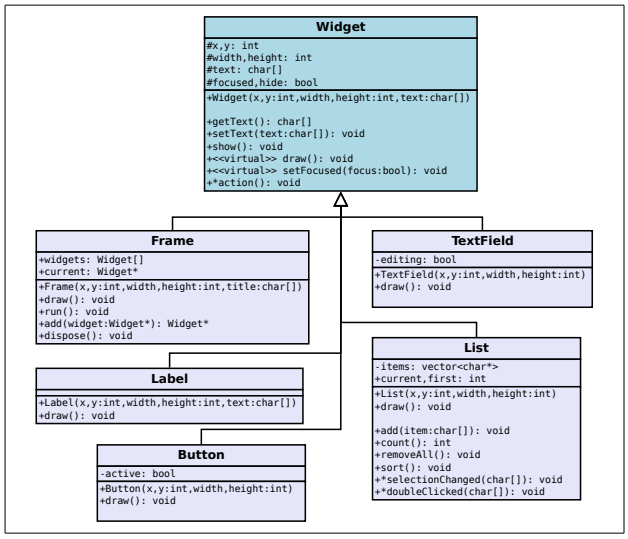
Para este fin se retomó un proyecto realizado con anterioridad, el cual utiliza el Sistema X para crear estos elementos por medio de programación orientada a objetos. Este proyecto se denominó "SimpleGUI". La última versión de esta aplicación se puede conseguir en: https://github.com/gonzalohernandez/simplegui.

Este proyecto fue iniciado con el objetivo académico de rehacer toda la base de objetos requeridos en un ambiente visual de programación. La familia de clases "Widget" puede apreciarse en la figura 4, donde se ha modelado algunas de las clases.

En su implementación se han utilizado las primitivas de dibujo, como líneas, textos y colores para crear cada uno de los objetos mencionados, todas las clases tienen sobrecargado el metodo polimórfico "draw()" encargado de esta función gráfica.

Para proveer el servicio de programación de eventos, se han creado métodos de tipo apuntador. Esta característica les permite ser implementados en forma tardía al momento de asignar su dirección de memoria a un método posterior. Por ejemplo, el método void Widget::*action(); será utilizado para los eventos comunes, tales como click sobre un botón o un campo de texto.

Figura 4. Familia Widget



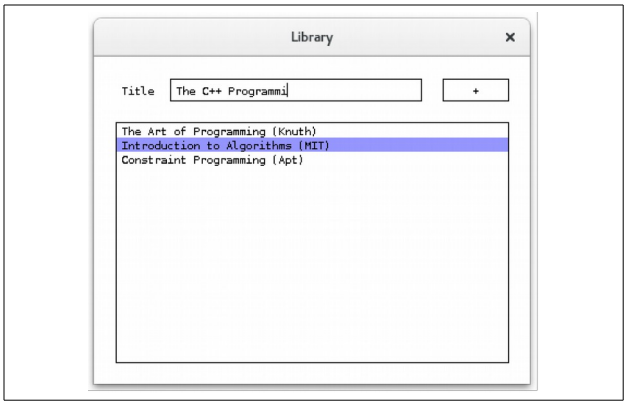
En la figura 5, se presenta un sencillo código utilizando la librería "SimpleGUI" para crear una ventana que permita adicionar títulos de libros a un listado. La ejecución se vería como lo muestra la figura 6 después de haber agregado algunos títulos.

Figura 5. Librería simple.cpp

```
#include <simplegui.h>
using namespace std ;
class Library ;
Library * library ;
class Library : public Frame {
private :
   Label* info ;
   TextField * title ;
```

```
   Button* insert ;
   List* store ;
public :
   Library()
      : Frame(100,100,400,300,
      "Library") {
      info = new Label(20,20,50,20,
         "Title" );
      title = new TextField (70,20,
         230,20);
      insert = new Button (320,20,
         60,20, "+" );
      store = new List (20,60,
         360,220);
      add(info);
      add(title);
      add(insert);
      add(store);
      library = this;
      insert -> action = & addBook;
      run ();
   }
   static void addBook() {
      library->store->add(
         library ->title->getText());
      library->title->setText("");
   }
};
int main() {
   Library l ;
}
```

Figura 6. Ejecución de simple.cpp



Después de revisar la pertinencia de utilizar como base el proyecto SimpleGUI, se determinó que sí es viable hacer un "Branch" al proyecto en GitHub y completar su desarrollo utilizando nuevas primitivas de dibujo ofrecidas por OpenGL gracias a las cuales el ambiente puede ser modelado en tres dimensiones.

2.3 Implementación de la tecnología SBS 3D

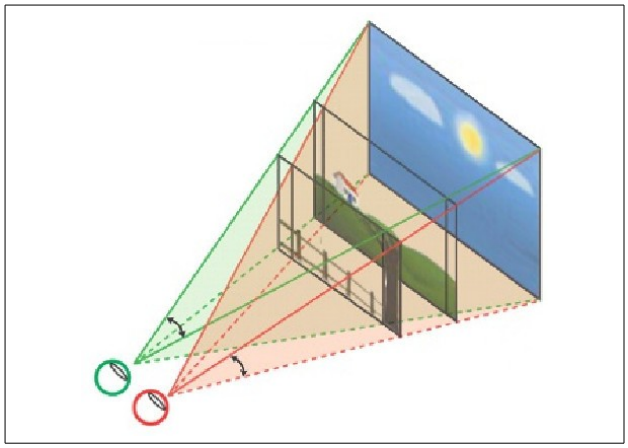
La proyección de video en forma estereoscópica permite que un usuario pueda ver la profundidad de los objetos sobre una pantalla de dos dimensiones. Esto se logra gracias a que el video es emitido en dos proyecciones diferentes sobre la misma escena, una para cada uno de los ojos del usuario [9].

La proyección que le corresponde al ojo izquierdo deberá recibir una presentación de la escena con el punto de vista desplazado ligeramente a la izquierda, y por el contrario, la proyección que le corresponde al ojo derecho recibirá una presentación de la escena con el punto de vista desplazado ligeramente a la derecha. La figura 7 bosqueja este proceso.

El cerebro se encarga de procesar las dos imágenes y crear el efecto de profundidad de los objetos que se encuentra en la escena.

Actualmente, es posible conseguir tanto contenido estereoscópico [5], como dispositivos para verlos [4] [3] [11]. La tecnología que se utiliza para lograr este propósito consiste en fusionar los dos videos (left + right) en uno solo, donde cada frame esta dividido en dos zonas, las figura 8 muestra las dos técnicas de mezclado.

Figura 7. Vista estereoscópica



En el estudio realizado para determinar la forma en que se creará el escritorio estereoscópico, se determinó utilizar SBS (side-by-side) [17], los monitores 3D que se venden en el comercio son capaces de extraer los dos videos y enviarlos al usuario en diferentes frecuencias.

Figura 8. Técnica Side-By-side vs Over-under

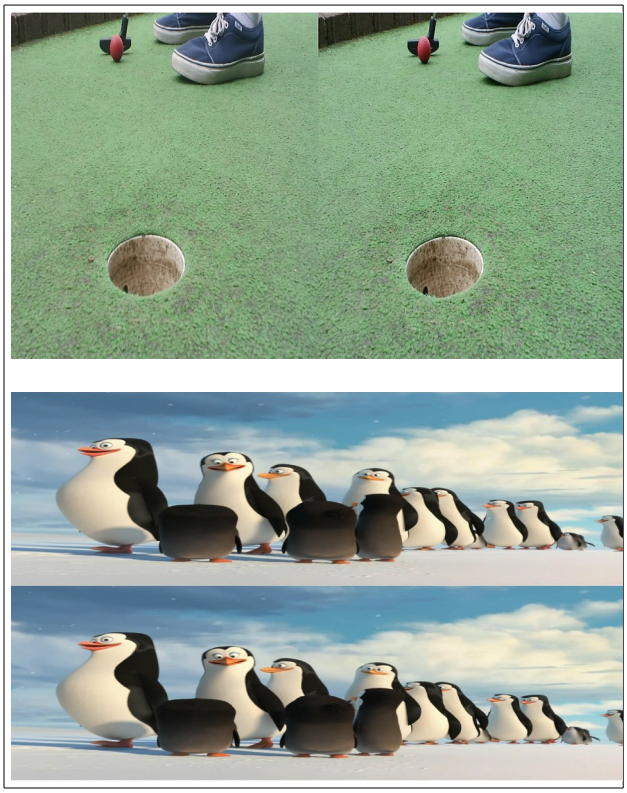
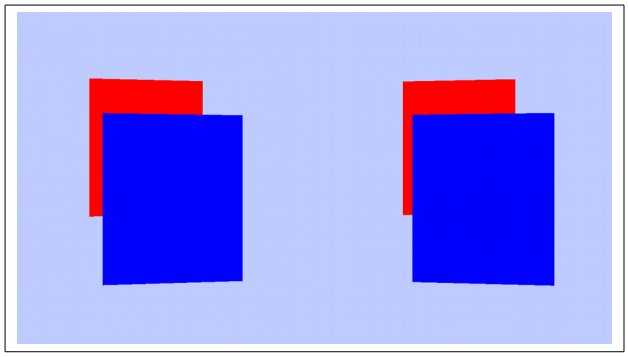


Figura 9. Dos Frames



2.4 Usando el API OpenGL para proporcionar profundidad.

Para crear el escritorio estereoscópico se crearán las dos perspectivas del ambiente, graficando sobre X11, dibujando con las primitivas de dibujo de OpenGL [13] y ubicando dos cámaras para extraer dos videos diferentes [6] [7]. La figura 9 muestra la ejecución de dos rectángulos, uno rojo y otro azul (simulando dos ventanas), la escena ya tiene los dos videos unidos para que posteriormente el monitor 3D los separe y emita en diferentes frecuencias cada uno de los lados.

En la página oficial del proyecto http://sonar.udenar.edu.co/s-desktop se puede ver el código completo con el cual se crearon los dos rectángulos, en dicho código se puede ver que están ubicados en diferentes profundidades (-50.0f y -30.0f). Para crear el video SBS se han ubicado dos cámaras separadas a -3.0 y 3.0 unidades del centro de la presentación [2][13].

3. CONCLUSIONES Y TRABAJOS FUTUROS

Hasta el momento se ha demostrato que es posible la construcción de un escritorio esteroscópico utilizando El Systema X, La librería SimpleGUI, El Mezclado de video estereoscópico SBS y el API de graficación 3D OpenGL.

Para soportar este escritorio se require como hardware un PC con una tarjeta aceleradora de video, una pantalla 3D o un televisor 3D y gafas filtro pasivas.

Como resultados del avance de este Proyecto se cuenta con el "Branch" del repositorio del proyecto SimpleGUI y se esta adecuando para poder dibujar con las primitivas de dibujo de OpenGL, la generación de los dos videos se logrará ubicando dos camaras en la misma escena con diferentes posiciones.
Como trabajos futuros se cuentan aquellas actividades relacionadas con la culminación del proyecto, tales como:

El puntero del mouse sera deshabilitado y se creará un icono propio que se mueva junto con el del ratón. El movimiento deberá ser ajustado para que solo trabaje en la mitad izquierda de la pantalla y que genere un duplicado en la zona derecha, de esta forma el puntero también adquirirá las características de profundidad.

En la investigación aun se esta diseñando la forma en la que se manipulará el mouse para que el usuario también decida la profundidad en la que el puntero se moverá. En este sentido se esta estudiando el documento de la doctora Shemali [12].

En general queda como trabajo futuro completar la segunda fase del proyecto que consiste en terminar el prototipo del escritorio estereoscópico, con el cual se pueda hacer pruebas de funcionalidad y hacer algunas implementaciones que se proponen en [14] y en [15].

En una tercera etapa se analizará el comportamiento de los usuarios frente al escritorio estereoscópico, lo cual permitirá obtener mas conclusiones sobre el proyecto.

4. REFERENCIAS

[1] Archlinux. Desktop environment. https://wiki.archlinux.org/index.php/Desktop_environment, May 2015.

[2] Donal Hearn M. Pauline Baker. Gráficos por computadora con OpenGL. Pearson Education, Madrid, 3 edition, 2004.

[3] Sony Corporation. 3d tvs. http://www.sony.com/electronics/3d-tvs, 2016.

[4] LG Electronics. Startling realistic 3d. http://www.lg.com/us/3d-tvs, 2016.

[5] Torsten Hoffmann. 3d content hub. http://3dcontenthub.com/, 2015.

[6] Yongjin Kim. Stereoscopic 3d line drawing. In ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings, volume 32, 2013.

[7] Yunjin Lee. Binocular depth perception of stereoscopic 3d line drawings. In Proceedings of the ACM Symposium on Applied Perception, pages 31–34, 2013.

[8] O'Reilly. Xlib Programming Manual. O'Reilly Associates, Inc, United States, 1 edition, 1992.

[9] Robert Earl Patterson. Human factors of 3d displays. In Handbook of Visual Display Technology, pages 1815–1822, 2012.

[10] James Gettys Robert W. Scheifler. X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD. Digital Equipment Corporation, United States, 1990.

[11] Samsung. 3d active glasses. http://www.samsung.com/us/video/tvs-accessories/SSG-5150GB/ZA, 2016.

[12] Leila Schemali. Design and evaluation of mouse cursors in a stereoscopic desktop environment. In 3D User Interfaces (3DUI), 2014 IEEE Symposium on, pages 67–70, 2014.

[13] Graham Sellers. OpenGL SuperBible: Comprehensive Tutorial and Reference. Addison-Wesley, United States, 5 edition, 2014.

[14] Takashi Shibata. Stereoscopic 3-d display with optical correction for the reduction of the discrepancy between accommodation and convergence. Journal of the Society for Information Display,13:665–671, 2005.

[15] Andy van Dam. Ieee computer graphics and applications. IEEE Computer Society, (62), 1981-2013.

[16] Wikipedia. Xlib. https://en.wikipedia.org/wiki/Xlib, Dec 2015.

[17] DVB, "Frame compatible plano-stereoscopic 3DTV," DVB Document A154, Feb. 2011.