Artículo **Revista de Sistemas Computacionales y TIC's**
Marzo 2017 Vol.3 No.7 1-6

Interfaz gráfica en WPF XNA para monitoreo de control de Xbox 360

ABRIL-GARCÍA, José Humberto†*, MEZA-IBARRA, Iván Dostoyewski, CHENOWETH-CHENOWETH Ivan Rogelio

Universidad Tecnológica de Hermosillo

Recibido Septiembre 22, 2016; Aceptado Febrero 09, 2017

Resumen

Se plantea el desarrollo de una interfaz gráfica de usuario en WPF y XNA para monitoreo de un control de Xbox 360, con el objetivo de diseñar un programa que permita capturar y mostrar en pantalla los botones presionados en el control, como base para el desarrollo de aplicaciones más complejas y robustas. El desarrollo se realizó en C# .Net utilizando Visual Studio 2015 Community Edition y el framework XNA. Esta aplicación permite visualizar en tiempo real las acciones en pantalla cuando el usuario presiona los botones del control, las cuales se despliegan en la aplicación con lo que se puede observar y monitorear las acciones realizadas sobre él. Con este proyecto se logra un guía de desarrollo que cualquier persona puede utilizar para interactuar con este tipo de dispositivos para desarrollar aplicaciones en el área de las Tecnologías de la Información y la Comunicación como videojuegos, control industrial, automatización y mecatrónica, entre otros.

GUI, WPF, XNA, Xbox 360, Control

Abstract

This is a proposal of development of a graphical user interface in WPF and XNA, to monitor an Xbox 360 control, with the objective to design a program that allows capturing and displaying on the screen the buttons pressed in the control, as a basis for development to reach more complex and robust applications. The development was coded in C# .Net using Visual Studio 2015 Community Edition and the XNA framework. This application allows visualizing in real time the actions on the screen when the user presses the buttons of the control, which are displayed in the application to observe and monitor the actions performed on it. This project provides a development guide that anyone can use to interact with this type of device for the applications development in the area of information technologies and communication, such as video games, industrial control, automation and mechatronics, among others.

GUI, WPF, XNA, Xbox 360, Control

Citación: ABRIL-GARCÍA, José Humberto, MEZA-IBARRA, Iván Dostoyewski, CHENOWETH-CHENOWETH Ivan Rogelio. Interfaz gráfica en WPF XNA para monitoreo de control de Xbox 360. Revista de Sistemas Computacionales y TIC'S 2017, 3-7: 1-6

*Correspondencia al Autor (Correo Electrónico: abril@uthermosillo.edu.mx)
†Investigador contribuyendo como primer autor.

Introducción

El desarrollo y comercio actual de videojuegos es una de las industrias más lucrativas, inclusive sobre el cine y televisión, por esta razón es de gran importancia para cualquier desarrollador conocer las bases de la programación de videojuegos para al menos una consola a motor de videojuegos, por lo que consideramos relevante proponer el desarrollo de esta aplicación que muestra una GUI donde se visualiza un control de Xbox 360 (Microsoft, 2016) y que una vez detectado por la aplicación despliega en pantalla los botones y joystick que se presionen en el control, que sea de utilidad para los que deseen desarrollar aplicaciones más complejas.

Para el desarrollo de este proyecto se utiliza Visual Studio 2015 (© 2016 Microsoft, 2016) y el framework XNA, el cual se puede obtener descargando el instalador sin costo de la página oficial. El resultado que se obtiene es esencia una guía de desarrollo, que puede ser usada en la creación de aplicaciones con mayor alcance en múltiples áreas de la ingeniería como una base para la programación de videojuegos o aplicaciones para múltiples plataformas que soportan el framework XNA (Microsoft, Microsoft XNA Game Studio 4.0, 2016). La figura 1 muestra el diagrama general del proyecto.

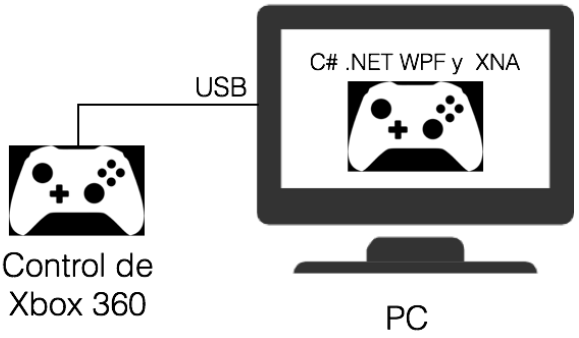

Figura 1 Diagrama general del proyecto. Elaboración propia

Referentes Teóricos

La empresa Microsoft con la creciente demanda del desarrollo para video juegos, lanzó desde el 2006 XNA Game Studio 4.0, es cual es un entorno de programación que le permite usar el popular IDE Visual Studio para crear juegos en las diferentes plataformas de Microsoft como: Windows Phone, Xbox 360 console, y sistemas operativos de Windows.

XNA Framework facilita en sus librerías muchas funcionalidades ya integradas y especializadas para elaborar videojuegos, ofreciendo al desarrollador despreocuparse de muchas cuestiones técnicas específicas. Es decir, no es necesario dedicar tiempo al desarrollo en el manejo de memoria, gráficos, sonidos, controladores, temporizadores o en optimización, si no al comportamiento del juego. Es decir, que el desarrollo se enfoca en el diseño y del comportamiento del juego en sí, y no encargarnos de otros aspectos más técnicos, por lo que se ahorra tiempo en la codificación.

También incorpora el seguimiento continuo encargado de importar, compilar y cargar el código con poco esfuerzo, en las diferentes plataformas, haciendo uso de librerías multiplataforma de audio y video, para juegos en 2D o 3D, además en el uso de la variedad de controles y accesorios de Microsoft.

El framework de XNA está estructurado en una serie de capas en su arquitectura, en la cual la plataforma es la capa más baja, y está compuesta por las APIs sobre las que está construida XNA. Algunas de estas APIs son Direct3D de video, XACT para audio, XInput para controles y X para el Núcleo del framework, donde el núcleo es la primera capa de XNA que nos proporciona el acceso a características y directivas agrupadas por su funcionalidad como: Audio, Gráficos, Datos Entrada, Almacenamiento y Matemáticas (Figura 2).

ISSN-2444-5002

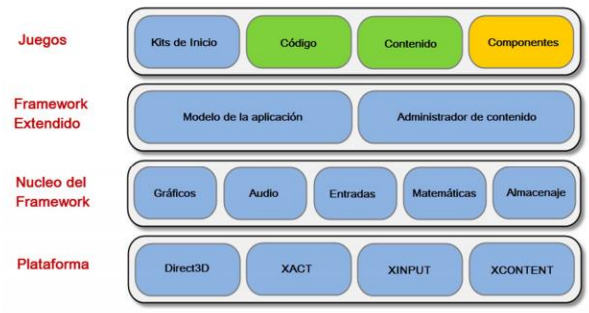

Figura 2 Capas y funcionalidades de XNA Framework. Elaboración propia

Desarrollo del proyecto

A continuación, listamos las herramientas utilizadas para el desarrollo del proyecto.

Hardware:

- Computadora Intel(R) Core(TM) i5-5257 CPU @2.70 GHz 2.70GHz, 8 GB (RAM), procesador x64.
- Control alámbrico de Xbox 360 modelo 52A-00004

Software:

- Windows 8.1 Pro
- Microsoft Visual Studio Community 2015 Version 14.0.25424.00 Update 3
- DirectX
- XNA Framework 4.0 Redistribution
- XNA Game Studio 4.0 Platform Tools
- XNA Game Studio 4.0 Shared
- XNA Game Studio 4.0.vsix

Una vez instalado todo el software se inicia como un proyecto nuevo en VS 2015. El tipo de aplicación a seleccionar es Xbox 360 Game (4.0) (ver Figura 3), esto solo será posible solamente si se instalan las herramientas mencionadas anteriormente.

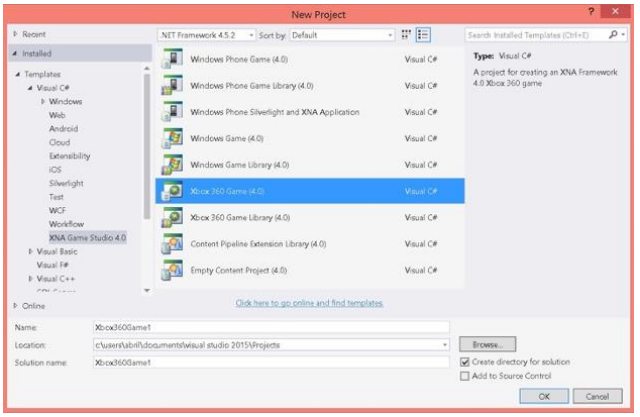

Figura 3 Pantalla de creación de una aplicación para Xbox 360 Game. Elaboración propia

La diferencia en utilizar WPF a WinForms es que no se utilizan Threads (hilos) para el monitoreo del control, haciendo que mejore el rendimiento de la aplicación y la optimización de recursos. La desventaja es, que no se cuenta con un modo de diseño, pero el rendimiento en la ejecución compensa esta desventaja.

En la Figura 4 observamos que el proyecto solo crea dos archivos: program.cs conteniendo el método main y Game1.cs que hereda de la clase Microsoft.Xna.Framework.Game, en ninguno de los dos permite el modo de diseño.

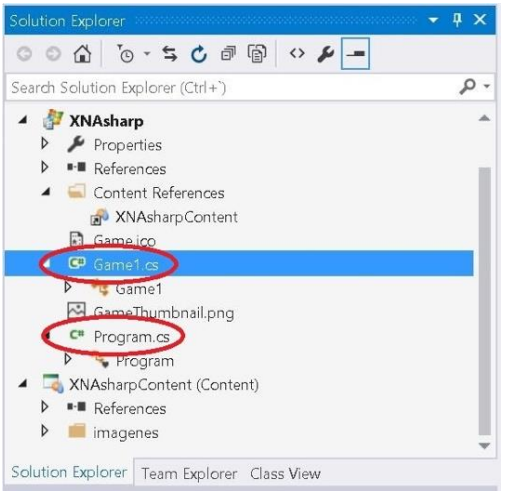

Figura 4 Archivos generados por Visual Studio con el código de la aplicación. Elaboración propia

Una vez que el diseño de la interfaz es finalizado, se procede a detectar la correcta instalación y conexión del control. Se codifica la captura de los eventos usando C#, en este caso se captura los botones del control, y finalmente se integra todo el proyecto. El programa procede a mostrar en pantalla los botones presionados en el control, esto se visualiza resaltando la imagen del botón presionado, y mostrando de manera independiente una gráfica que hace referencia al botón presionado.

En la Figura 5 observamos la aplicación operando, y como está presionado en el control el botón "B", el cual es marcado con un indicador azul y adicionalmente se muestra en la esquina inferior izquierda del GUI.

Figura 5 GUI mostrando en botón presionado físicamente en el control. Elaboración propia

Análisis del código

El código fuente fue elaborado (Anexo) con base al ciclo de vida de un juego en XNA framework (Figura 6). Usando solo la clase del proyecto principal llamada Game1, la cual contiene el método constructor Game1() definiendo el tamaño de la ventana que tiene la interfaz gráfica con dimensiones de 800x500 pixeles; el método de Initialize() , donde se asignan las posiciones de cada elemento gráfico; ValidaBtn() activa o desactiva cada botón y el vibrador del control XBOX; LoadContent() carga las imágenes; Update() valida actualizaciones de los botones; y el método Draw() que se encarga de dibujar los gráficos dentro de la ventana dependiendo del valor de los botones presionados.

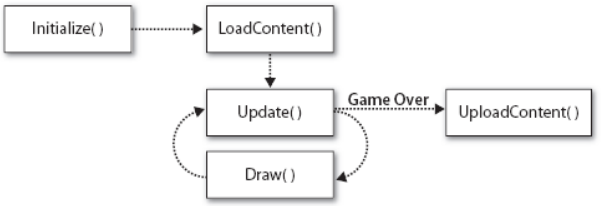

Figura 6 Ciclo de vida de XNA Framework Elaboración propia

Resultados y conclusiones

En la figura 6 vemos el resultado final de nuestro proyecto, donde se encuentra conectado y operando el control de Xbox 360 con la interface visual.

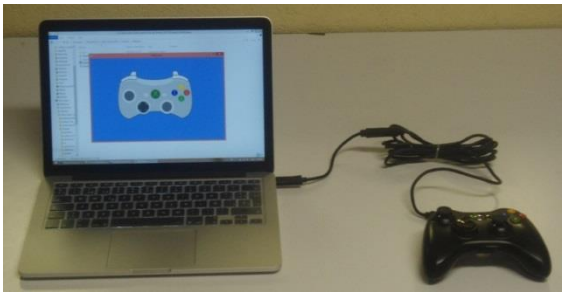

Figura 6 Resultado final. Elaboración propia

Con el desarrollo del presente trabajo logramos generar una aplicación base para Xbox 360 y compatible con cualquier plataforma que soporte el Framework XNA, la cual nos ha servido de guía para poder incursionar en desarrollos y creación de video juegos, en las diferentes plataformas en las que opera XNA, donde los estudiantes pueden obtener muchos de los beneficios de esta plataforma de desarrollo, entre los cuales podemos mencionar las posibilidades de distribución que nos facilita Microsoft para nuestra creaciones, además de la integración de las herramientas y librerías, para diseñar, programar y probar nuestros juegos, dedicando el tiempo en las características más relevantes del comportamiento y funcionalidades del video juego, dejando de lado aspectos como el rendimiento, la administración, seguridad y la potencia del software, ya que esto es gestionado por el Framework XNA.

ISSN-2444-5002

También se logró conocer las bases del diseño de videojuegos y del desarrollo en estas plataformas. Esta aplicación puede ser usada en el desarrollo de proyectos más complejos y robustos, como la creación de simulaciones, sistemas de control, o en la integración de proyectos multidisciplinarios como la educación, mecatrónica etc.

Anexos

Código fuente de la aplicación.

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
namespace XNAsharp
{
public class Game1 : Microsoft.Xna.Framework.Game
{
      GraphicsDeviceManager graphics;
      SpriteBatch spriteBatch;
      Vector2 controlPos, barridoPos;
      Texture2D controlTex, boton;
      List<Vector2> botonesPos;
      List<Texture2D> botonesTex;
      bool vibracion = false;
      bool [] botonesActivos;
       Texture2D botonA, botonB, botonX, botonY,
// …Sucesivamente  el resto de botones …
       public Game1()
      {
       graphics = new GraphicsDeviceManager(this);
       graphics.PreferredBackBufferWidth = 800;
       graphics.PreferredBackBufferHeight = 500;
       Content.RootDirectory = "Content";
} // end of Game1 ()
protected override void Initialize()
{
       controlPos = new Vector2(160, 80);
      barridoPos = new Vector2(0, 400);
      botonesPos = new List<Vector2>();
      botonesPos.Add(new Vector2(525, 233));
// … Sucesivamente para el resto de posisiones...
      botonesTex = new List<Texture2D>();
      botonesActivos = new bool[19];
      base.Initialize();
} // end of Initialize()
protected override void LoadContent()
{
       spriteBatch = new
SpriteBatch(GraphicsDevice);
       controlTex =
       Content.Load<Texture2D>("imagenes/ControlXbox
       ");
// …Sucesivamente  el resto de las imágenes ...
       foreach (var boton in botonesPos)
             botonesTex.Add(botonATex);
} // end of LoadContent()
protected void validadBtn(ButtonState GPX, Texture2D
botonx, int i)
{
      if (GPX == ButtonState.Pressed){
                botonesActivos[i] = true;
            if (vibracion)
GamePad.SetVibration(PlayerIndex.One, 0.5f, 1.0f);
                boton = botonx;
       }
} // end of validaBtn()
protected override void Update(GameTime gameTime)
{
      for (int i = 0; i < botonesActivos.Length;
      i++){
          botonesActivos[i] = false;
      }
      GamePadState control =
      GamePad.GetState(PlayerIndex.One);
      if (control.IsConnected){
      validadBtn(control.Buttons.A, botonA, 0);
// … Susesivamente para el resto botones …
if (control.Triggers.Right == 1f){
      botonesActivos[4] = true;
      if (vibracion)
GamePad.SetVibration(PlayerIndex.One, 0.5f, 1.0f);
      boton = botonRT;
      }
      if (control.Triggers.Left == 1f){
      botonesActivos[7] = true;
                  if (vibracion)
GamePad.SetVibration(PlayerIndex.One, 0.5f, 1.0f);
                  boton = botonLT;
               }
               if (control.ThumbSticks.Left.X !=
0.0f | control.ThumbSticks.Left.Y != 0.0f){
                  botonesActivos[17] = true;
                  boton = botonLS;
                  if (vibracion)
GamePad.SetVibration(PlayerIndex.One, 0.5f, 1.0f);
               }
               if (control.ThumbSticks.Right.X !=
0.0f | control.ThumbSticks.Right.Y != 0.0f){
                  botonesActivos[18] = true;
                  boton = botonRS;
                  if (vibracion)
GamePad.SetVibration(PlayerIndex.One, 0.5f, 1.0f);
               }
            }
         }
         protected override void Draw(GameTime
gameTime)
         {
         GraphicsDevice.Clear(Color.CornflowerBlue);
         botonesPos[17] = new Vector2(214 +
(GamePad.GetState(PlayerIndex.One,
GamePadDeadZone.IndependentAxes).ThumbSticks.Left.X)
* 20,213 + (GamePad.GetState(PlayerIndex.One,
GamePadDeadZone.IndependentAxes).ThumbSticks.Left.Y)
* -20);
         botonesPos[18] = new Vector2(437 +
(GamePad.GetState(PlayerIndex.One,
GamePadDeadZone.IndependentAxes).ThumbSticks.Right.X)
* 20, 280 + (GamePad.GetState(PlayerIndex.One,
GamePadDeadZone.IndependentAxes).ThumbSticks.Right.Y)
* -20);
          spriteBatch.Begin();
          spriteBatch.Draw(controlTex, controlPos,
Color.White);
for (int i = 0; i < botonesPos.Count; i++)
       if (botonesActivos[i])
         spriteBatch.Draw(botonesTex[i],
botonesPos[i], Color.White);
        spriteBatch.End();
        spriteBatch.Begin();
```

ISSN-2444-5002

```
            for (int i = 0; i < botonesPos.Count;
            i++)
            if (botonesActivos[i])
            spriteBatch.Draw(boton, barridoPos,
            Color.White);
           spriteBatch.End();
           base.Draw(gameTime);
        }
    }
}
```

Referencias

Videojuego para el repaso de fracciones "Tséem Took y la princesa de Uxmal Versión 1.1". | Facultad de Matemáticas, Universidad Autónoma de Yucatán, México Revista: TE & ET; no. 14 | Diciembre 2014 | http://teyet-revista.info.unlp.edu.ar | ISSN 1850-9959 | RedUNCI-UNLP

Manejo de entornos virtuales mediante el sensor "Kinect" y su aplicación en fisioterapia. | IT Tuxtla Gutierrez | ISSN 2007-9400, www.revistatecnologiadigital.com

WEB VIRTUAL BALL GAME "POK-TA-POK" ISSN: 1992-8645 | www.jatit.org | E-ISSN: 1817-3195

© 2016 Microsoft. (2016). Visual Studio IDE. Recuperado el 03 de Septiembre de 2016, de Visual Studio IDE: https://beta.visualstudio.com/vs/

Microsoft, ©. 2. (2016). Microsoft XNA Game Studio 4.0. Recuperado el 2016 de Septiembre de 2016, de Microsoft XNA Game Studio 4.0: https://www.microsoft.com/en-us/download/details.aspx?id=23714

Microsoft, ©. 2. (2016). Xbox 360 Controller for Windows | Accessories de Microsoft. Recuperado el 03 de Septiembre de 2016, de Xbox 360 Controller for Windows | Accessories de Microsoft: https://www.microsoft.com/accessories/es-es/products/gaming/xbox-360-controller-for-windows/52a-00005

(2002). Ingeniería de Software, un enfoque práctico. En R. S. Pressman, Ingeniería de Software, un enfoque práctico. McGraw-Hill.