Anexos del trabajo: Incidencia de la realimentación háptica en el desempeño de la rehabilitación motriz de miembro superior - Caso de estudio

Alejandro Mejía Morales
Milton Hernán Arango Giraldo

Director: M.Sc.(c) Diego Enrique Guzmán Villamarin
Codirector: PhD. Carlos Felipe Rengifo Rodas

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, 2016

Anexos del trabajo: Incidencia de la realimentación háptica en el desempeño de la rehabilitación motriz de miembro superior - Caso de estudio

Alejandro Mejía Morales
Milton Hernán Arango Giraldo

Trabajo de grado presentado a la Facultad de Ingeniería
Electrónica y Telecomunicaciones de la
Universidad del Cauca para la obtención del Título de:
Ingeniero en Automática Industrial
Ingeniero en Electrónica y Telecomunicaciones

Director: M.Sc.(c) Diego Enrique Guzmán Villamarin
Codirector: PhD. Carlos Felipe Rengifo Rodas

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán, 2016

Índice general

Índice de figuras

Índice de códigos

APÉNDICE A

Guía de Instalación

A.1. Introducción

Esta guía de instalación muestra los pasos necesarios para realizar una correcta instalación de todos los componentes requeridos para el funcionamiento del entorno virtual de rehabilitación. Estos incluyen; ejecutable del juego, aplicación de gestión de pacientes, *drivers* del dispositivo háptico y la base de datos de pacientes.

A.2. Requerimientos del sistema

Para asegurar un rendimiento óptimo del entorno de rehabilitación virtual *"Smash It"*, se requiere tener en cuenta los siguientes requerimientos hardware y software:

Hardware

- Procesador Intel core i5 @2.5 GHz o AMD Equivalente (Recomendado: Intel Core i7 @3GHz).
- Memoria RAM 4 GB (Recomendado: 8GB).
- Espacio en el disco duro 420 MB.
- Pantalla con resolución de 1024 x 768 px.

Software

- Sistema operativo Windows 7 o superior (x64).

- Requiere permisos de instalación (Administrador).

A.3. Instalación del entorno virtual

Para realizar una instalación sencilla del entorno virtual se desarrollo un instalador que permite automatizar este proceso. Para instalar el entorno virtual ejecute el instalador llamado "setup.exe" dentro del directorio *Instalador*. Para instalar este archivo es necesario tener permisos de administrador en el sistema. Si no los tiene, debe consultar al administrador de el computador. En la figura A.1 se puede ver el instalador una vez ejecutado, si lo desea puede crear un icono en el escritorio para mayor accesibilidad. Se aclara, que las ventanas posteriores a la mostrada en la figura A.1 no contienen ningún tipo de información a seleccionar, sólo se debe presionar los botones "siguiente" hasta que se vislumbre el de "Finalizar".

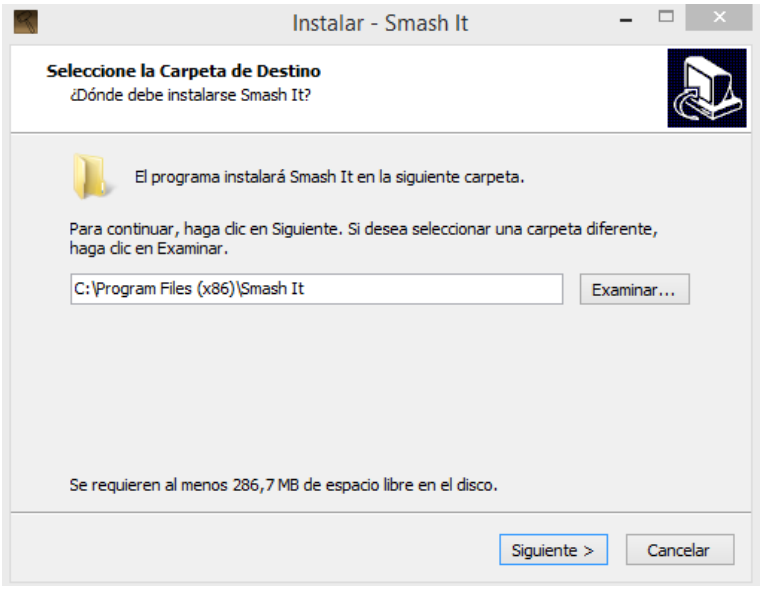

Figura A.1: Instalador del entorno virtual

Por otro lado, este instalador también cuenta con su correspondiente des-instalador, el cual genera los registros necesarios para eliminarlo del sistema. Si desea des-instalar el entorno virtual, lo puede hacer desde **Panel de control ->Programas y características ->Smash It ->Desinstalar**, este proceso eliminara todos los archivos creados en el proceso de instalación. Este proceso no puede ser revertido, si requiere usar el entorno virtual de nuevo, debe realizar la instalación nuevamente.

APÉNDICE **B**

Guía de configuración del dispositivo háptico en Unity3D

En este anexo se muestra el procedimiento para configurar el dispositivo *Omni Phantom* en el entorno Unity3D instalado en un computador de escritorio que cumpla con los requerimientos hardware listados en el apéndice A. Si se tiene un ordenador que cumpla con las características necesarias se puede proceder a realizar los pasos subsecuentes, es importante mencionar que es absolutamente necesario configurar el dispositivo háptico antes de ejecutar el entorno virtual, de lo contrario no funcionará.

B.1. Instalación de la tarjeta Firewire

Para poder conectar el dispositivo háptico al computador, es necesario contar con las entradas que requiere el *Phantom Omni* las cuales son puertos *IEEE 1394a Firewire*, en el mercado se encuentra una gran variedad de opciones para adquirir una tarjeta *Firewire*, en la figura B.1 se puede ver una de estas tarjetas. Además es necesario utilizar un cable IEEE 1394 de 6 pines con conectores “Macho a Macho”.

Para poder instalar la tarjeta en el computador es necesario tener un puerto **PCI** en la tarjeta madre del computador. Para instalar la tarjeta se deben seguir los pasos ilustrados en la guía de instalación del fabricante de la tarjeta, la cual por lo general viene dentro de la caja de la misma o está disponible en linea. Después de adaptarla se deben instalar los controladores de dicha tarjeta en el computador para que este la reconozca, éstos controladores se encuentran en el CD del producto o pueden ser encontrados en linea por el asistente software de Windows.

Figura B.1: Tarjeta IEEE 1394a Firewire [Tomado de [1]]

B.2. Instalación de controladores del Omni Phantom

Una vez realizada la instalación de la tarjeta Firewire, se procede a instalar los controladores necesarios para la configuración del dispositivo háptico en la aplicación. Para realizar la instalación de estos controladores se deben utilizar los archivos provistos, los cuales pueden verse en la figura B.2.

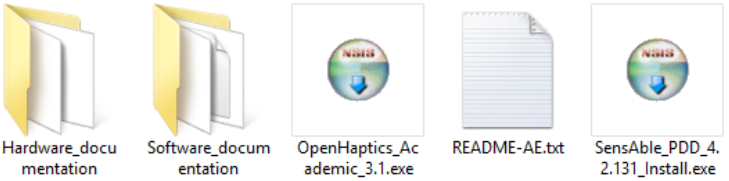

Figura B.2: Archivos de instalación del controlador

En la figura anterior se puede ver el directorio que contiene los archivos de instalación del *OpenHaptics SDK* y del *PHANTOM Device Driver* (PDD). Ambos archivos deben instalarse para poder utilizar el dispositivo háptico. A continuación se enumeran los pasos para realizar la instalación de estos controladores. Cabe mencionar que el proceso de instalación de estos archivos es similar a un instalador genérico y no requiere configuraciones adicionales por parte del usuario.

1. Desinstale cualquier versión anterior del *OpenHaptics SDK* y del *PHANTOM Device Driver*.
2. Ejecute el instalador del controlador "SensAble PDD 4.2.131 Install.exe". Asegúrese de instalar todos los componentes que vienen dentro del instalador, en la figura B.3 se puede ver cómo deben quedar seleccionados los componentes. Además, Si el programa solicita que debe reiniciar el computador, debe reiniciarlo. Una vez lo reinicie el dispositivo debería ser reconocido por el computador. En el escritorio puede encontrar un icono llamado "Phantom Test" el cual le permitirá calibrar el dispositivo.
3. A continuación debe ejecutar el instalador "OpenHaptics Academic 3.1.exe" el cual contiene las librerías necesarias para desarrollar aplicaciones que requieran utilizar dispositivos hápticos como el *Phantom Omni*. En la carpeta también puede encontrar documentación adicional. Una vez completados los pasos anteriores el dispositivo está listo para utilizarse.

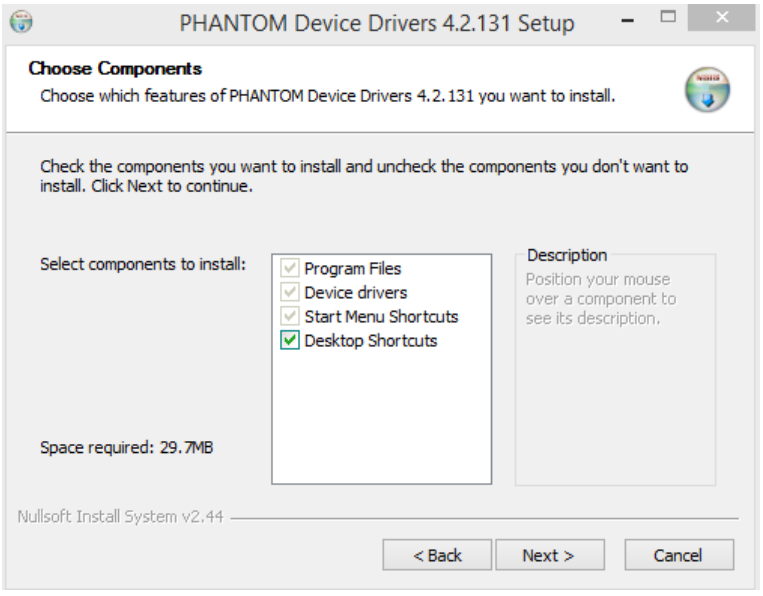

Figura B.3: Instalador de controladores para el *Omni Phantom*

B.3. Instalación del dispositivo *Phantom Omni* en el entorno de desarrollo software Unity3D

Una vez instalado en el ordenador los controladores mencionados anteriormente tal que el dispositivo háptico sea reconocido por el mismo, se procede a la instalación de la interfaz háptica en el entorno de desarrollo Unity3D. Se aclara, que para dicha instalación el entorno de desarrollo software debe estar también instalado en el ordenador. La documentación de éste entorno puede verificarse vía web [2], específicamente la información acerca de la instalación del mismo se puede ver en [3]. Además, en [4] se expone las características principales de la interfaz del programa y proporciona al usuario una pequeña familiarización con dicho entorno.

A continuación, se enumeran los pasos a seguir para la instalación básica del dispositivo *Phantom Omni* en el entorno de trabajo Unity3D, para lo cual se basará en [5].

1. Iniciar un nuevo proyecto en el entorno de desarrollo software Unity3D.

2. Mediante [6, 7] importar el proyecto *"Unity 5 Haptic Plugin for Geomagic Open Haptics 3.3 (HLAPI/HDAPI)"* de la *asset store* al proyecto nuevo que se creó en el paso anterior. Para verificar que éste paso se ha cumplido, en la ventana de proyecto de Unity3D (ver [4]) debe aparecer una carpeta con el nombre *"Haptic Project Components"*.

3. Copiar en la carpeta raíz del proyecto actual de Unity3D los siguientes archivos: *glut32.dll*, *hd.dll* y *hl.dll*. Para éste caso en particular, el primer archivo se encuentra en el directorio **OpenHaptics//Academic//3.1//utilities//lib//x64//ReleaseAcademicEdition** y el segundo y tercer archivo se encuentran en **OpenHaptics//Academic//3.1//lib//x64//ReleaseAcademicEdition** del ordenador. La ubicación de la carpeta OpenHaptics varía de acuerdo en dónde se haya instalado éste controlador, ver sección B.2. Se aclara que usualmente, la carpeta *OpenHaptics* queda instalada en la unidad **C:** del ordenador utilizado.

   Por otro lado, la carpeta raíz del proyecto actual es aquella donde está ubicada la carpeta *Assets* de dicho proyecto. Para llegar a ella de una manera sencilla, basta con dar click derecho a la carpeta *Assets* en la ventana de proyecto de Unity3D y posteriormente seleccionar la opción *Shown in Explorer*, lo que sucede a continuación es que se abre en el navegador de ventanas de Windows la ubicación de la carpeta *Assets*. En éste punto, se navega hacia la carpeta que contiene a *Assets* y es aquí donde deben ser copiados los archivos anteriormente encontrados. En la figura B.4 se puede apreciar una captura de

pantalla de cómo deben quedar los archivos mencionados en el proyecto creado del paso 1.

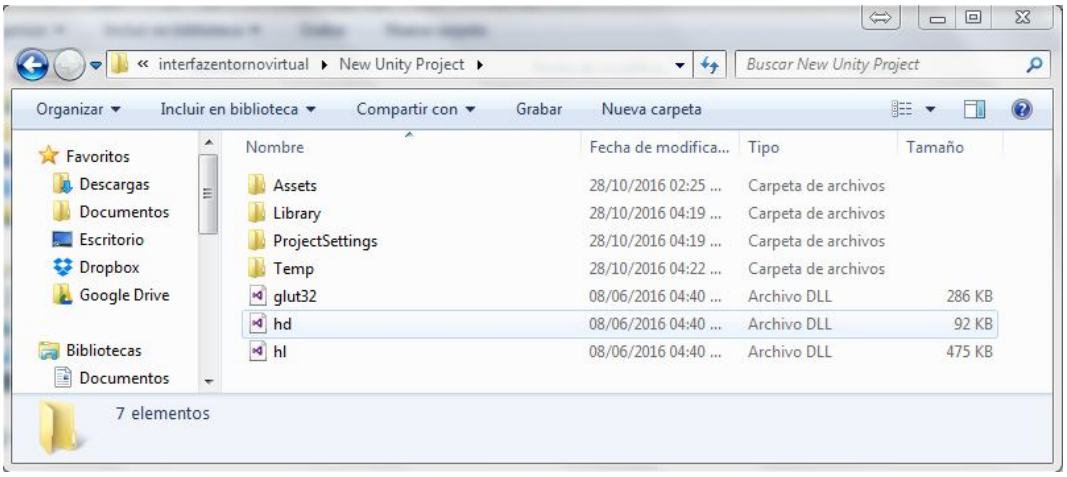

Figura B.4: Librerías en el directorio raíz del proyecto creado

4. Mover la carpeta *Plugins* que está dentro de la carpeta *Haptic Project Components* (nombrada en el paso 1) a la carpeta *Assets* del proyecto creado. En la figura B.5 se puede apreciar el resultado visto desde la ventana de proyecto de Unity3D.

5. Abrir la dirección **Assets//Haptic Project Components//Scene** y seleccionar el archivo *"Basic_Writing.unity"*. Una vez en la escena de vista de Unity3D se cargue el archivo, buscar el botón *play* en la barra de herramientas de Unity3D y presionarlo. A continuación, se debe cargar una escena predefinida por la Escuela de Arte de Glasgow en la que el usuario puede escribir en un tablero virtual, si esto no sucede y por el contrario Unity3D arroja un error, indica que el proceso anterior se hizo mal y que este debe ser repetido.

Se aclara que en caso de presentarse alguna dificultad propia a la secuencia de pasos dada anteriormente, se puede consultar en [5] información inherente a éste proceso.

B.4. Configuración del dispositivo *Phantom Omni* en el entorno de desarrollo software Unity3D

Una vez instalado el dispositivo háptico en el entorno de desarrollo Unity3D, se puede iniciar la elaboración libre y abierta de entornos gráficos que utilicen la interfaz háptica *Phantom*

Figura B.5: Carpetas del proyecto visto desde la ventana de proyecto de Unity3D

Omni. A continuación, se ahondará en dos características de configuración esenciales para la elaboración de ambientes virtuales que hacen uso de la interfaz háptica *Phantom Omni* según el esquema dado por la Escuela de Arte de Glasgow, se hace énfasis en que la explicación dada a continuación no definirá elementos y funcionalidades primitivas del entorno de desarrollo Unity3D, por lo cual, se recomienda fomentar una base de conceptos sobre dicho entorno en su respectivo manual disponible en [2].

Cabe aclarar, que ambas configuraciones se tomarán de un ejemplo predefinido por la Escuela de Arte de Glasgow, éste es *"Custom_Force_Effect.unity"* y se puede encontrar en la carpeta *Scene* del proyecto creado. Además, es apropiado enfatizar que el entorno de programación Unity3D es un entorno de programación orientado a objetos y orientado a eventos, por lo cual se podrá apreciar un manejo fácil e intuitivo de los *scripts* hápticos. También se define que un *script* no es más que un segmento de código que puede ser asociado como objeto de una clase a cualquier objeto virtual del escenario, y que un objeto virtual fundamental en Unity3D es considerado como un *"gameobject"*.

B.4.1. Configuración del escenario virtual

- Todos los escenarios virtuales tienen un *gameobject* llamado *"dummy"*. Éste es muy importante ya que se compone de los *scripts* que indican los efectos hápticos a utilizar en el escenario en general (por ejemplo, un escenario con efecto de fuerza constante o con efecto de viscosidad), se compone además del *script* que indica la modalidad en que se va a utilizar el dispositivo háptico (recordar que son cuatro modalidades y éstas están

definidas en el capítulo dedicado a la interfaz háptica en la monografía) y posee además un *script* genérico que contiene todas las funciones respecto a dicho dispositivo háptico, por lo general, éste último no es modificado por el usuario.

- El *script* que indica la modalidad de juego puede situar al dispositivo háptico en cuatro modalidades distintas (contacto simple, manipulación de objetos, efecto de fuerza personalizado, o punción), dependiendo de la modalidad escogida el *script* será distinto, éstos *scripts* están ubicados en la locación **Assets//Haptic Project Components//Scripts//Script Scene**. En la figura B.6 se puede apreciar los *scripts* disponibles para las distintas modalidades además de otros concernientes a la configuración de la escena del entorno virtual.

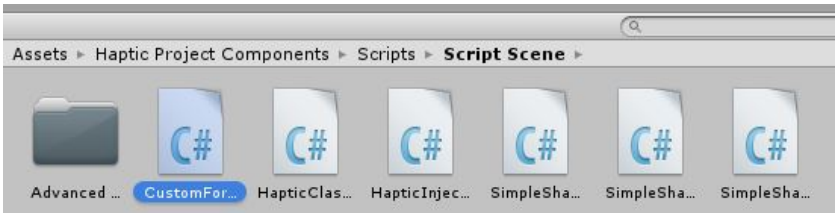

Figura B.6: *Scripts* para la configuración del escenario virtual háptico

- El escenario *Custom_Force_Effect.unity* configura el dispositivo háptico en la modalidad efecto de fuerza personalizado, ya que éste escenario utiliza los efectos de fuerza constante, de viscosidad, de vibración, de fuerza tangencial y de vibración de contacto. Esto se puede corroborar en el hecho de que el *gameobject "dummy"* de dicho escenario, tiene asociado el *script "CustomForceEffect.cs"*.

  El *script "CustomForceEffect.cs"* permite la simulación de distintos efectos hápticos, entre estos efectos está la sensibilidad de las geometrías presentes en el escenario. Sin embargo, éste *script* lo que hace es cargar inicialmente todas las geometrías de los objetos virtuales y posteriormente las simula por medio la interfaz. Lo anterior presenta desventajas si se desea generar un ambiente virtual que esté modificando la posición de las geometrías de los objetos virtuales, tal caso se presentó por ejemplo, en la elaboración del subsistema "realidad virtual" del sistema virtual-háptico del trabajo de pregrado anterior. En éste, se deseaba aplicar el concepto del ejercicio *"Whack a Mole"*, el cual exige que los objetos del entorno virtual varíen de posición a lo largo del tiempo.

En B.1 se puede apreciar el código *CustomForceEffect.cs* utilizado en el desarrollo del sistema virtual-háptico del presente trabajo de pregrado. Éste a diferencia del código original, permite la actualización de geometrías a lo largo del tiempo y su modificación principal se puede encontrar en la línea de código número 66. En ésta se dice que por cada evento de actualización, se le re-establecerá al dispositivo háptico todas las geometrías presentes en el entorno virtual.

Código B.1: *Script "CustomForceEffect.cs"* para escenarios virtuales que necesitan la actualización de geometrías de objetos virtuales que varían su posición o dirección a lo largo del tiempo

```
1  using UnityEngine;
2  using System.Collections;
3  using System;
4  using System.Runtime.InteropServices;
5
6  public class CustomForceEffect : HapticClassScript {
7
8  //Generic Haptic Functions
9  private GenericFunctionsClass myGenericFunctionsClassScript;
10
11 void Awake()
12 {
13 myGenericFunctionsClassScript = transform.GetComponent<GenericFunctions
       Class>();
14 }
15
16 void Start()
17 {
18 if(PluginImport.InitHapticDevice())
19         {
20                 Debug.Log("OpenGL Context Launched");
21                 Debug.Log("Haptic Device Launched");
22
23                 myGenericFunctionsClassScript.SetHapticWorkSpace();
24                 myGenericFunctionsClassScript.GetHapticWorkSpace();
25
26                 //Update Workspace as function of camera
27                 PluginImport.UpdateWorkspace(myHapticCamera.transform.r
                       otation.eulerAngles.y);
28
29                 PluginImport.SetMode(ModeIndex);
```

```
30                  //Show a text descrition of the mode
31                  myGenericFunctionsClassScript.IndicateMode();
32
33                  //Set the touchable face(s)
34                  PluginImport.SetTouchableFace(ConverterClass.ConvertStr
                        ingToByteToIntPtr(TouchableFace));
35
36          // Viscous Force Example
37          myGenericFunctionsClassScript.SetEnvironmentViscosity();
38
39          // Constant Force Example - We use this environmental force eff
                ect to simulate the weight of the cursor
40          myGenericFunctionsClassScript.SetEnvironmentConstantForce();
41
42          //Custom Force Effect Vibration Motor
43          myGenericFunctionsClassScript.SetVibrationMotor();
44
45          //Custom Force Effect Vibration at Contact//Good for plulsation
46          myGenericFunctionsClassScript.SetVibrationContact();
47
48          //Custom Tangential Force corresponding to that of a rotating p
                ower tool (e.g. Drill, Polisher, Grinder)
49          myGenericFunctionsClassScript.SetTangentialForce();
50
51          myGenericFunctionsClassScript.SetHapticGeometry();
52
53          PluginImport.LaunchHapticEvent();
54      }
55      else
56      {
57          Debug.Log("Haptic Device cannot be launched");
58      }
59  }
60
61
62  void Update()
63  {
64      if (true)
65      {
66          myGenericFunctionsClassScript.SetHapticGeometry();
67
68          PluginImport.UpdateWorkspace(myHapticCamera.transform.rotation.
                eulerAngles.y);
```

```
69
70          myGenericFunctionsClassScript.UpdateGraphicalWorkspace();
71
72          PluginImport.RenderHaptic();
73
74          myGenericFunctionsClassScript.GetProxyValues();
75
76          myGenericFunctionsClassScript.GetTouchedObject();
77          }
78  }
79
80  void OnDisable()
81  {
82          if (PluginImport.HapticCleanUp())
83          {
84                  Debug.Log("Haptic Context CleanUp");
85                  Debug.Log("Desactivate Device");
86                  Debug.Log("OpenGL Context CleanUp");
87          }
88  }
89  }
```

- En la ventana *Inspector* del entorno Unity3D (ver [4]) es configurado al *gameobject "dummy"* en el campo del *script* que controla la modalidad (en éste caso *"Custom Force Effect"*) el espacio de trabajo de la interfaz háptica dentro de todo el escenario virtual, el objeto que representará al lápiz háptico y la cámara (vista del usuario) para dicho dispositivo háptico. En la figura B.7 se puede apreciar la vista de la ventana *Inspector* anteriormente mencionada, se puede apreciar los campos en los cuales se configura las variables anteriormente mencionadas.

- El espacio de trabajo que se mencionó en el ítem anterior hace referencia al campo de acción que tiene el dispositivo háptico con respecto a todo el espacio virtual presente en el entorno 3D. Este espacio viene definido por las configuraciones hápticas predeterminadas de la Escuela de Arte de Glasgow (específicamente en el script *"GenericFunctionsClass.cs"*) y no están presentadas de manera tal que, el usuario pueda configurarlas a su gusto. En la elaboración del sistema háptico-virtual de éste trabajo de pregrado se necesitó ampliar este espacio de trabajo para que cubriese todo el escenario virtual diseñado ya que originalmente, el rango efectivo de acción del dispositivo *Phantom Omni* no cubría todo el diseño planteado. Por éste motivo, tuvo que ser modificado en el código fuente de los *scripts* de la

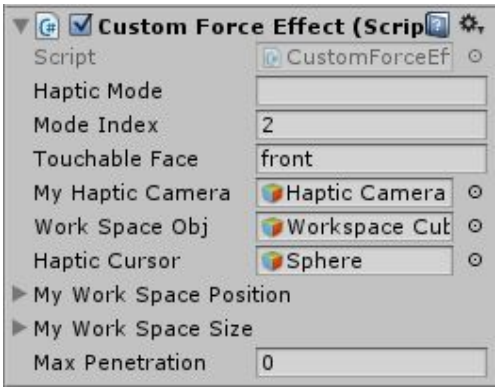

Figura B.7: Vista del script *"CustomForceEffect.cs"* asociado al *gameobject "dummy"* en la ventana inspector de Unity3D

interfaz háptica. En el código B.2 se muestra la variante al método *"GetProxyValues()"* y en el código B.3 la variante al método "SetHapticGeometry()", ambos pertenecientes al *script "GenericFunctionsClass.cs"*, para duplicar el espacio de trabajo original ofrecido por la interfaz háptica utilizada.

Según lo visto en el código fuente de la interfaz háptica, se puede apreciar que los cálculos de las fuerzas son proporcionadas por el dispositivo mismo. En otras palabras, el funcionamiento de todo el sistema es el siguiente; La interfaz háptica es configurada inicialmente por el entorno virtual Unity3D bajo una modalidad y unos efectos de fuerza específicos, luego, la interfaz proporciona a Unity3D la ubicación actual del lápiz háptico según su referencia interna que ya está calibrada, posteriormente, Unity3D envía constantemente al dispositivo háptico la ubicación de las distintas geometrías que tiene presentes en el entorno virtual, esto es, la malla de puntos que configuran esas geometrías. El dispositivo háptico por consiguiente toma todas las geometrías y las analiza con respecto a la posición de lápiz háptico, consecuentemente, la interfaz crea todos los efectos correspondientes al caso particular que se le presente en un momento dado.

Con base a la descripción dada anteriormente, se puede apreciar que la modificación principal del método *GetProxyValues()* está en la línea de código 12, la cual simplemente se traduce en que la posición que está dando la interfaz háptica del lápiz sea multiplicada por dos, de esta manera, un movimiento del lápiz en el mundo físico real implicaría un movimiento duplicado en el mundo virtual. De ésta misma forma, se aprecia que las variantes al método *SetHapticGeometry()* se encuentran de la línea de código 16 a la 19, éstas

modificaciones se traducen en que Unity3D enviará todas los puntos de las mallas de los objetos virtuales en una posición duplicada a la que está representada en el mundo virtual, de ésta manera, un objeto en determinada posición virtual sería visto por la interfaz háptica en un punto duplicado en distancia. Con ambas modificaciones en el funcionamiento descrito del párrafo anterior, se establece que el rango efectivo de la interfaz háptica en el ambiente virtual ha sido duplicado.

Código B.2: Variante del método *"GetProxyValues()"* del *script "GenericFunctions-Class.cs"* para un espacio de trabajo duplicado con respecto al espacio de trabajo original ofrecido por el código fuente de la interfaz háptica

```
1  //Get Proxy Position and Orientation generic function
2  public void GetProxyValues()
3  {
4          //Convert IntPtr to Double3Array
5          myProxyPosition = ConverterClass.ConvertIntPtrToDouble3(PluginI
               mport.GetProxyPosition());
6
7          //Attach the Cursor Node
8          Vector3 positionCursor = new Vector3();
9          positionCursor = ConverterClass.ConvertDouble3ToVector3(myProxy
               Position);
10
11         //Assign Haptic Values to Cursor
12         myHapticClassScript.hapticCursor.transform.position = positionC
               ursor*2;
13
14     //Proxy Direction
15         //Convert IntPtr to Double3Array
16         myProxyDirection =  ConverterClass.ConvertIntPtrToDouble3( Plug
               inImport.GetProxyDirection());
17         //Attach the Cursor Node
18         Vector3 directionCursor = new Vector3();
19         directionCursor = ConverterClass.ConvertDouble3ToVector3(myProx
               yDirection);
20         //Proxy Torque
21         myProxyTorque = ConverterClass.ConvertIntPtrToDouble3(PluginImp
               ort.GetProxyTorque());
22         //Attach the Cursor Node
23         Vector3 torqueCursor = new Vector3();
24         torqueCursor = ConverterClass.ConvertDouble3ToVector3(myProxyTo
               rque);
```

```
25
26          //Set Orientation
27          myHapticClassScript.hapticCursor.transform.rotation = Quaternio
                n.LookRotation(directionCursor,torqueCursor);
28  }
```

Código B.3: Variante del método *"SetHapticGeometry()"* del *script "GenericFunctions-Class.cs"* para un espacio de trabajo duplicado con respecto al espacio de trabajo original ofrecido por el código fuente de la interfaz háptica

```
1   public void SetHapticGeometry()
2   {
3           //Get array of all object with tag "Touchable"
4           GameObject[] myObjects = GameObject.FindGameObjectsWithTag("Tou
                chable") as GameObject[];
5
6           for (int ObjId = 0; ObjId < myObjects.Length; ObjId++)
7           {
8                   //Set the Transformation Matric of the Object
9
10                  //Get the Transformation matrix from object
11                  Matrix4x4 m = new Matrix4x4();
12
13                  //Build a transform Matrix from the translation/rotatio
                        n and Scale parameters fo the object - Glabal Matri
                        x
14                  m  = myObjects[ObjId].transform.localToWorldMatrix;
15
16          m[3, 0] = m[3, 0] * 2;
17          m[3, 1] = m[3, 1] * 2;
18          m[3, 2] = m[3, 2] * 2;
19          m[3, 3] = m[3, 3] * 2;
20
21          //Convert Matrix4x4 to double16
22          double[] matrix = ConverterClass.ConvertMatrix4x4ToDouble16(m);
23                  //Convert Double16 To IntPtr
24                  IntPtr dstDoublePtr = ConverterClass.ConvertDouble16ToI
                        ntPtr(matrix);
25
26                  //Convert String to Byte[] (char* in C++) and Byte[] to
                         IntPtr
27                  IntPtr dstCharPtr = ConverterClass.ConvertStringToByteT
                        oIntPtr(myObjects[ObjId].name);
```

```
28
29                  //Send the transformation Matrix of the object
30                  PluginImport.SetObjectTransform(ObjId, dstCharPtr, dstD
                        oublePtr);
31
32                  //Set the Mesh of the Object
33
34                  //Get Mesh of Object
35                  Mesh mesh = myObjects[ObjId].GetComponent<MeshFilter>()
                        .mesh;
36                  Vector3[] vertices = mesh.vertices;
37
38                  int[] triangles = mesh.triangles;
39
40                  //Reorganize the Array
41                  float[] verticesToSend = ConverterClass.ConvertVector3A
                        rrayToFloatArray(vertices);
42                  //Allocate Memory according to needed space for float*
                        (3*4)
43                  IntPtr dstVerticesArrayPtr = Marshal.AllocCoTaskMem(ver
                        tices.Length * 3 * Marshal.SizeOf(typeof(float)));
44                  //Copy to dstPtr
45                  Marshal.Copy(verticesToSend,0,dstVerticesArrayPtr,verti
                        ces.Length * 3);
46
47                  //Convert Int[] to IntPtr
48                  IntPtr dstTrianglesArrayPtr = ConverterClass.ConvertInt
                        ArrayToIntPtr(triangles);
49
50                  //Send the Raw Mesh of the object - transformation are
                        not applied on the Mesh vertices
51                  PluginImport.SetObjectMesh(ObjId,dstVerticesArrayPtr, d
                        stTrianglesArrayPtr,vertices.Length,triangles.Lengt
                        h);
52
53                  //Get the haptic parameter configuration
54                  ReadHapticProperties(ObjId, myObjects[ObjId]);
55          }
56  }
```

- Tal como se hizo alusión en el primer ítem de ésta sección, un entorno virtual puede poseer diversos efectos hápticos. Estos son configurados en el *gameobject "dummy"* y

son asociados como objetos a éste. De lo anterior se tiene que cada efecto háptico será representado por un *script* específico, que configura todas las variables asociadas a dicho efecto y por ende, si se desea más de un efecto en un determinado entorno, entonces a éste *gameobject* le deberá ser agregado más de un *script* que los represente. Cabe aclarar que todos los *scripts* de los efectos hápticos están ubicados en la dirección **Assets//Haptic Project Components//Scripts//Force Effects**. En la figura B.8 se puede apreciar dos *scripts* inherentes a los efectos hápticos aplicados de manera predeterminada al *gameobject "dummy"* del escenario *Custom_Force_Effect.unity* en la ventana inspector de Unity3D.

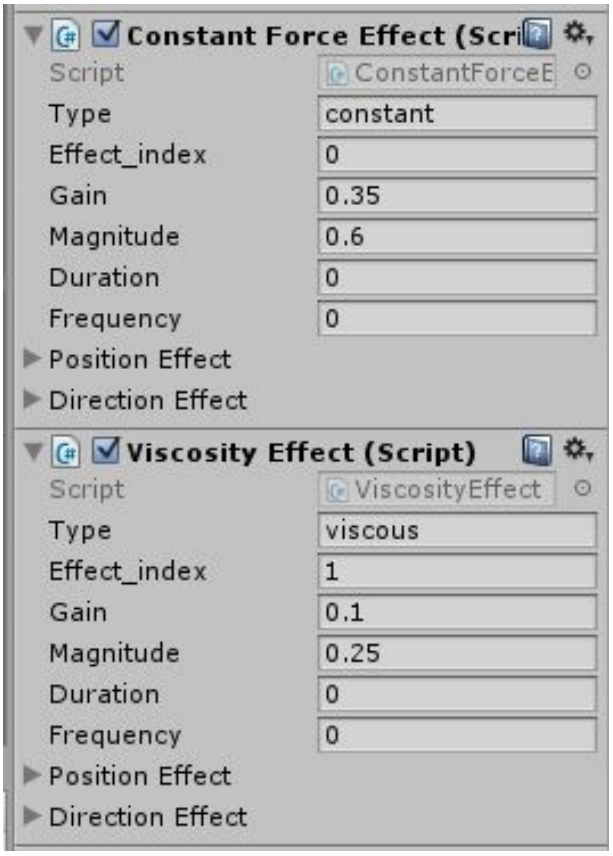

Figura B.8: *scripts* inherentes a los efectos hápticos aplicados de manera predeterminada al *gameobject "dummy"* del escenario *Custom_Force_Effect.unity* en la ventana inspector de Unity3D

B.4.2. Configuración de los objetos virtuales

- Como se mencionó anteriormente, el entorno Unity3D envía la malla de los objetos virtuales a la interfaz *Phantom Omni*, para que esto sea posible, todo objeto virtual que deba interactuar con dicha interfaz debe tener asociada la propiedad *"Mesh Filter"*. La propiedad *"Mesh Filter"* asocia a un objeto (*gameobject*) una nube de puntos que conforman mediante polígonos la estructura virtual del mismo. En la figura B.9 se puede apreciar la vista de la ventana inspector de Unity3D del objeto *"Mesh Filter"* asociado a un *gameobject* que representa una esfera virtual.

Figura B.9: Objeto *"Mesh Filter"* de un *gameobject* visto desde la ventana inspector de Unity3D

- Para que un objeto virtual interactúe con la interfaz háptica, éste debe ser clasificado de tal manera que pueda ser diferenciado de los objetos virtuales comunes. De lo contrario, el entorno no tendría conocimiento de qué información enviar a la interfaz háptica. Para ésto, cada objeto virtual a interactuar debe poseer la *Tag "Touchable"*. Para modificar la *Tag* de un objeto virtual basta con ubicarse en la parte superior de la ventana inspector de dicho objeto y seleccionar dicha *Tag* deseada, en la figura B.10 se puede apreciar la *Tag "Touchable"* para un *gameobject* del escenario *Custom_Force_Effect.unity*. En [8] se puede apreciar toda la información relacionada con las *Tags* en el entorno de desarrollo Unity3D.

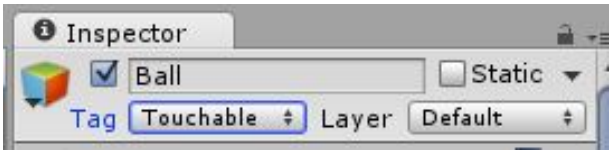

Figura B.10: *Tag* de un *gameobject* vista en la ventana inspector de Unity3D

- Las características hápticas de un objeto (descritas en el capítulo dedicado a la interfaz háptica *Phantom Omni*) son configuradas en el *script "HapticProperties.cs"*. Estas características permiten la elaboración de distintas texturas en el ambiente virtual, y pueden ser vistas en la figura B.11, en donde se muestra desde la venta inspector de Unity3D el *script "HapticProperties.cs"* asociado a un *gameobject* del escenario *Custom_Force_Effect.unity*.

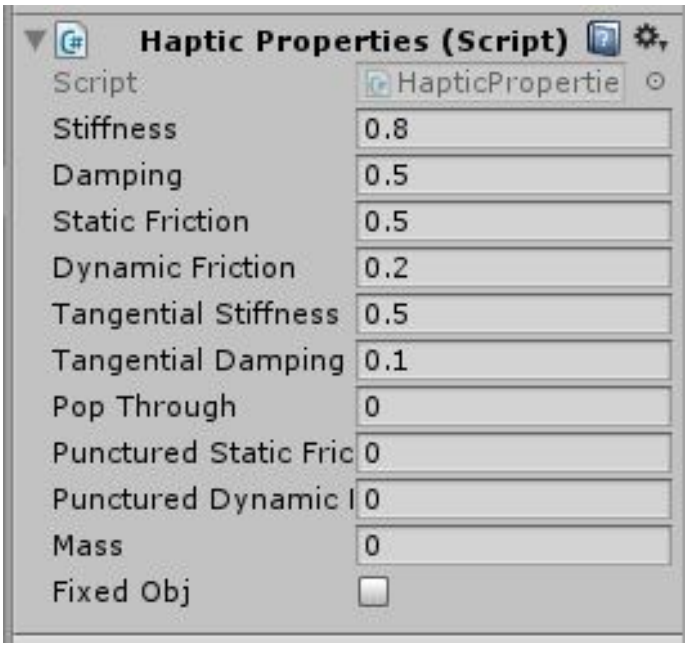

Figura B.11: *script "HapticProperties.cs"* asociado como objeto a un *gameobject* del escenario *Custom_Force_Effect.unity*

B.5. Personalización de las propiedades hápticas en un entorno virtual definido

Como se mencionó en la sección anterior, una vez instalado y configurado el dispositivo háptico *Phantom Omni* en el entorno de desarrollo software Unity3D se puede proceder libremente al diseño e implementación de entornos virtuales con la interfaz háptica utilizada en éste estudio. A diferencia de la sección anterior, aquí se pretende dar una base al lector para la personalización de variables hápticas de los entornos virtuales creados en Unity3D, se enfatiza el hecho de que una implementación de éste tipo puede ser llevada a cabo de distintas maneras, sin embargo, se empleará un ejemplo corto basado en el desarrollo del sistema virtual-háptico para éste estudio.

En ésta sección, se entiende como personalización cualquier mecanismo o conjunto de propiedades asignadas a un determinado sistema por un usuario que, en uso de su intelecto las crea o modifica bajo su criterio. La personalización de características hápticas en entornos virtuales desarrollados mediante Unity3D depende mucho de las habilidades en programación y creatividad que tenga el diseñador, pudiendo éste añadir entre otras propiedades el cambio aleatorio a lo largo del tiempo de los ejes de una fuerza constante aplicada a un ambiente virtual.

Ya que en el sistema virtual-háptico desarrollado para éste trabajo de pregrado se implementó la característica del cambio aleatorio a lo largo del tiempo de los ejes de una fuerza constante aplicada al ambiente virtual diseñado, a continuación se explicará cómo se configuraron los *scripts* hápticos para la implementación de dicha característica siguiendo los *scripts* elaborados para tal fin.

La propiedad a desarrollar se basa fundamentalmente en el efecto háptico de fuerza constante, a la cual se desea que sus ejes de acción varíen a lo largo del tiempo. La solución empleada fue el uso de variables globales que comunicaran el *script* encargado de personalizar las propiedades hápticas con el *script* inherente a dicha propiedad, el cual es consultado posteriormente por el *script* que se comunica con la interfaz háptica.

El *script* encargado de personalizar las propiedades hápticas es el creado por el usuario y en éste, debe existir un segmento de código encargado de cambiar aleatoriamente el eje de la fuerza constante cada determinado tiempo. En el código B.4 se puede apreciar la sección de inicialización de variables y los métodos *Awake()*, *Start()* y *ChangeAxisForces()* del *script "SelectFeaturesByLevel.cs"*. Este *script* es el encargado de configurar todas las características del escenario virtual en el sistema virtual-háptico desarrollado, para éste caso sólo se muestran las componentes asociadas a las características deseadas. Como se puede apreciar, las variables globales son inicializadas, entre ellas, se encuentra el objeto del *script "GenericFuncionsClass.cs"* que se encargará de invocar la actualización de las variables modificadas. Posteriormente, se puede apreciar en el método *Awake()* el instanciamiento de la clase anteriormente mencionada. Consecuentemente, en el método *Start()* se evidencia la inicialización del método *ChangeAxisForces()* como elemento *InvokeRepeating*, esto quiere decir, que el método anteriormente mencionado se repetirá a lo largo de la ejecución del proyecto cada cierto tiempo establecido. Finalmente, se puede apreciar la definición del método *ChangeAxisForces()* que es el encargado de aleatorizar los ejes de la fuerza constante del escenario virtual e invocar como tal la función encargada de actualizar la interfaz háptica. En conclusión, la sinergia de todo el código hace que cada cierto tiempo, el sistema modifique el eje en el que se está ejecutando la fuerza constante de manera aleatoria e invoque la función encargada de actualizar dichos datos.

Código B.4: Segmento del *script "SelectFeaturesByLevel.cs"* encargado de aleatorizar los ejes de la fuerza constante aplicada al entorno virtual

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class SelectFeaturesByLevel : MonoBehaviour {
5
6  public static string  level = "easy";
7
8  public static bool    forcesActivated = false;
9  public static float   magnitudeForceSelected = 0.0F;
10 public static float   gainForceSelected = 0.0F;
11 public static float   frecuenceForceSelected = 0.0F;
12 public static float[] directionForceSelected = new float[3];
13
14 public static bool    changeAxisAutomatic = false;
15
16 public static GenericFunctionsClass myGenericFunctionsClassScriptT;
17
18 void Awake() {
19     myGenericFunctionsClassScriptT = transform.GetComponent<GenericFunction
           sClass>();  // Object that will update the Omni
20 }
21
22 /* Use this for initialization */
23 void Start () {
24     InvokeRepeating("ChangeAxisForces", 1, 3.75F);
25
26 }
27
28 void ChangeAxisForces()
29 {
30     if(level == "legend" || (level == "personalized" && changeAxisAutomati
           c))
31     {
32         int index = (int)Random.Range(1, 6);  // chose randomly axis
33         switch (index)
34         {
35             case 1:
36                 directionForceSelected[0] = -1F;
37                 directionForceSelected[1] = 0F;
38                 directionForceSelected[2] = 0F;
39                 break;
```

```
40              case 2:
41                  directionForceSelected[0] = 1F;
42                  directionForceSelected[1] = 0F;
43                  directionForceSelected[2] = 0F;
44                  break;
45              case 3:
46                  directionForceSelected[0] = 0F;
47                  directionForceSelected[1] = -1F;
48                  directionForceSelected[2] = 0F;
49                  break;
50              case 4:
51                  directionForceSelected[0] = 0F;
52                  directionForceSelected[1] = 1F;
53                  directionForceSelected[2] = 0F;
54                  break;
55              case 5:
56                  directionForceSelected[0] = 0F;
57                  directionForceSelected[1] = 0F;
58                  directionForceSelected[2] = -1F;
59                  break;
60              case 6:
61                  directionForceSelected[0] = 0F;
62                  directionForceSelected[1] = 0F;
63                  directionForceSelected[2] = 1F;
64                  break;
65          }
66      }
67
68      myGenericFunctionsClassScriptT.SetEnvironmentConstantForce();
69      myGenericFunctionsClassScriptT.SetEnvironmentViscosity();
70  }
```

Sin embargo, se puede apreciar en el código anterior que las variables modificadas (variables globales) propias a la personalización del entorno aún no están conectadas hacia el *script* encargado de contener la información de las mismas para posteriormente, ser llevadas a la interfaz háptica. Para éste caso, el *script* encargado de lo anterior es el asociado al efecto háptico que se desea modificar, en otras palabras, se hace referencia al *script "ConstantForceEffect.cs"* por tratarse de un efecto de fuerza constante. Este es modificado como se muestra en el código B.5, dicha modificación consiste esencialmente en que los parámetros que éste *script* almacena se estarán actualizando constantemente de las variables globales propias del *script* anterior (*"SelectFeaturesByLevel.cs"*). De ésta manera, las funciones invocadas (por el *script*

"SelectFeaturesByLevel.cs") del *script "GenericFunctionsClass.cs"* envían a la interfaz háptica la información almacenada en el *script "ConstantForceEffect.cs"*, el cual ha actualizado sus valores anteriormente del *script* encargado de personalizar las variables hápticas, esto es, el *script "SelectFeaturesByLevel.cs"*.

Código B.5: Variante al *script "ConstantForceEffect.cs"* que actualiza constantemente las variables internas del *script* encargado de personalizar las características hápticas del entorno virtual

```
1  using UnityEngine;
2  using System.Collections;
3  using System;
4  using System.Runtime.InteropServices;
5
6  public class ConstantForceEffect : MonoBehaviour {
7
8          public string Type;
9          public int effect_index;
10         public float gain;
11         public float magnitude;
12         public float duration;
13         public float frequency;
14         public float[] positionEffect =  new float[3];
15         public float[] directionEffect = new float[3];
16
17
18         // Use this for initialization
19         void Start () {
20                 Type = "constant";
21         }
22
23         // Update is called once per frame
24         void Update () {
25         if (SelectFeaturesByLevel.forcesActivated == true)
26         {
27             Type = "constant";
28
29             effect_index = 0;
30             gain = SelectFeaturesByLevel.gainForceSelected;
31             magnitude = SelectFeaturesByLevel.magnitudeForceSelected;
32             frequency = SelectFeaturesByLevel.frecuenceForceSelected;
33             directionEffect = SelectFeaturesByLevel.directionForceSelected;
```

```
34          }
35          else
36          {
37              Type = null;
38              gain = 0F;
39              magnitude = 0F;
40              directionEffect[0] = 0;
41              directionEffect[1] = 0;
42              directionEffect[2] = 0;
43          }
44          }
45  }
```

APÉNDICE C

Manual del Usuario

C.1. Aplicación de gestión de pacientes

En esta sección se muestra la utilización de la interfaz de gestión de pacientes. Haciendo una explicación de cada una de las funciones que posee la aplicación. En la figura C.1 se puede observar la ventana principal de la aplicación. En esta se puede ver los diferentes botones que proveen las funcionalidades de la aplicación. A continuación se describe cada una de ellas.

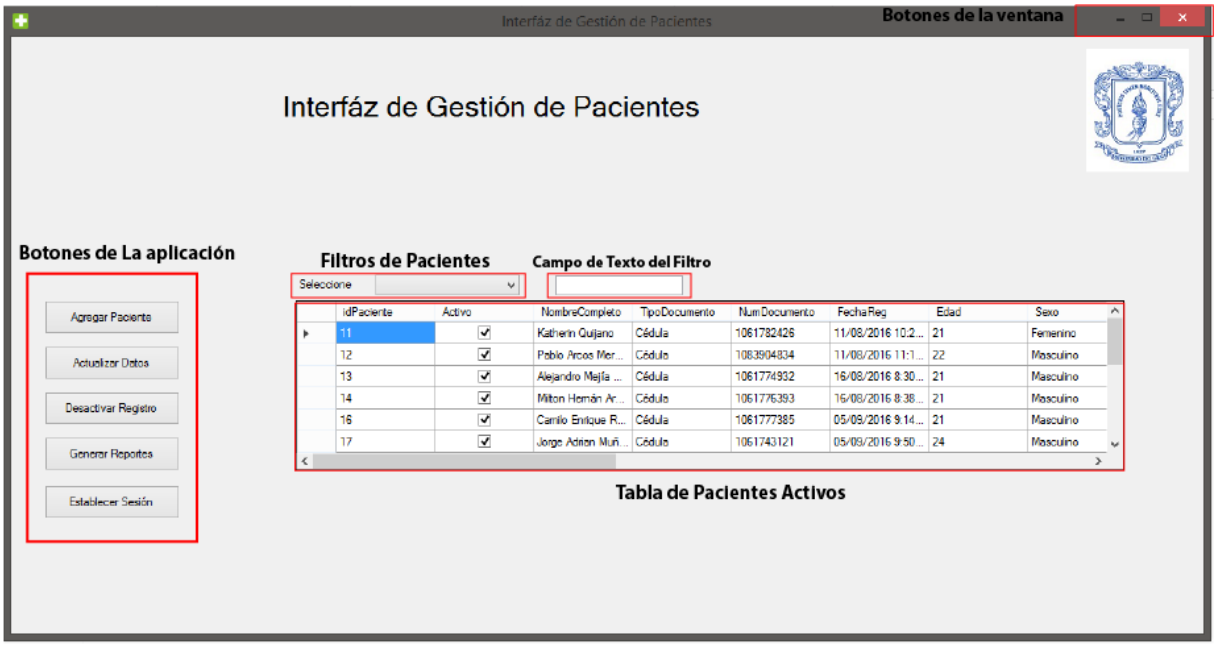

Figura C.1: Interfaz principal de la aplicación

C.1.1. Agregar paciente

Para registrar un nuevo paciente en la aplicación, debe hacer click en el botón “Agregar paciente”. Inmediatamente se abre una ventana que contiene el formulario para el registro de pacientes. En la figura C.2 se puede ver el formulario desplegado y los campos obligatorios, que en éste caso se muestran en el recuadro rojo. Los campos opcionales se pueden rellenar posteriormente.

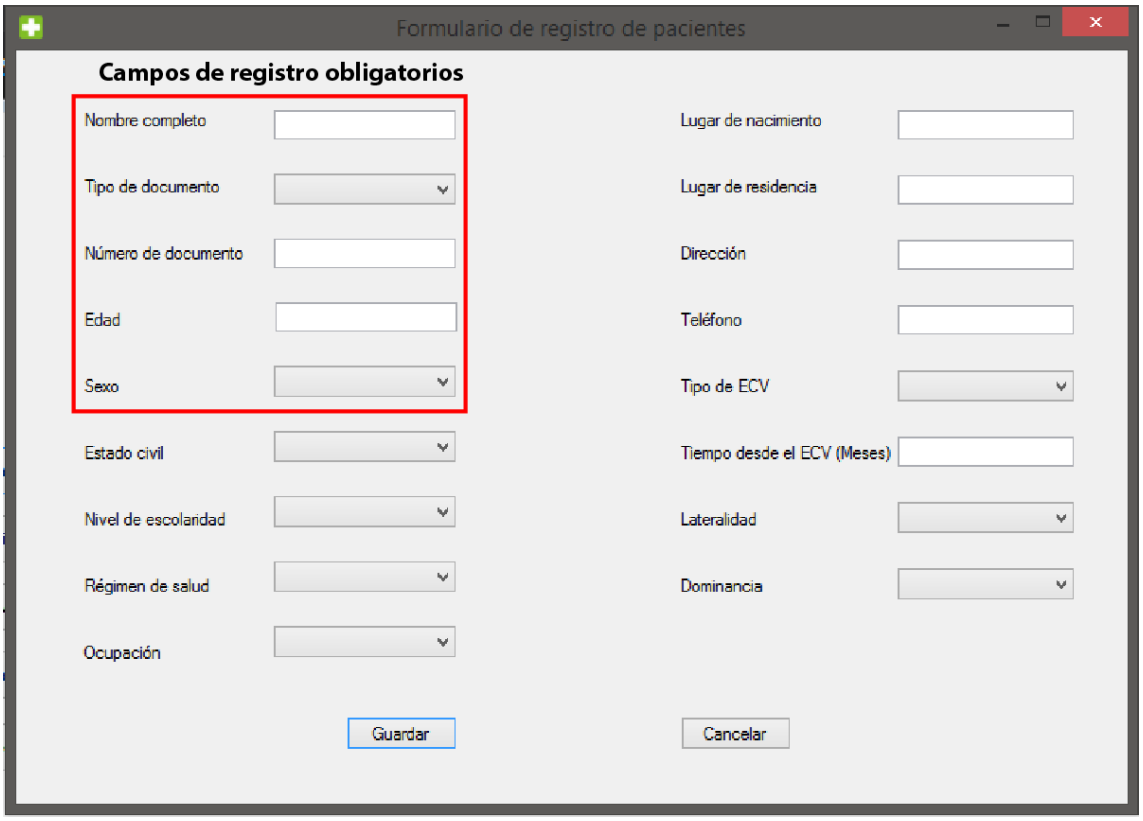

Figura C.2: Formulario de registro de pacientes

Finalmente, llene los datos obligatorios del paciente y haga click en el botón “Guardar” para almacenar los datos del paciente en la base de datos. Si no se cometieron errores, en la tabla de pacientes se puede ver que el último registro corresponde al paciente registrado previamente.

C.1.2. Actualizar datos

Cuando se requiera cambiar o actualizar los datos de un paciente se deben seguir los siguientes pasos:

1. Seleccione el paciente que desea actualizar haciendo click en la primera columna de la tabla, en la fila correspondiente a dicho paciente. Cuando toda la fila de dicho paciente este resaltada en color azul puede proceder a cambiar los datos que requiera.

2. Debe hacer doble click izquierdo sobre el campo que desea cambiar para poder habilitar el modo de edición de la tabla, puede verificar si se encuentra en modo edición si aparece un icono de un lápiz en la primera columna correspondiente al paciente resaltado actualmente.

3. Una vez terminadas las actualizaciones a los datos de un paciente. Debe hacer click en el botón “Actualizar Datos” para guardar los cambios en la base de datos.

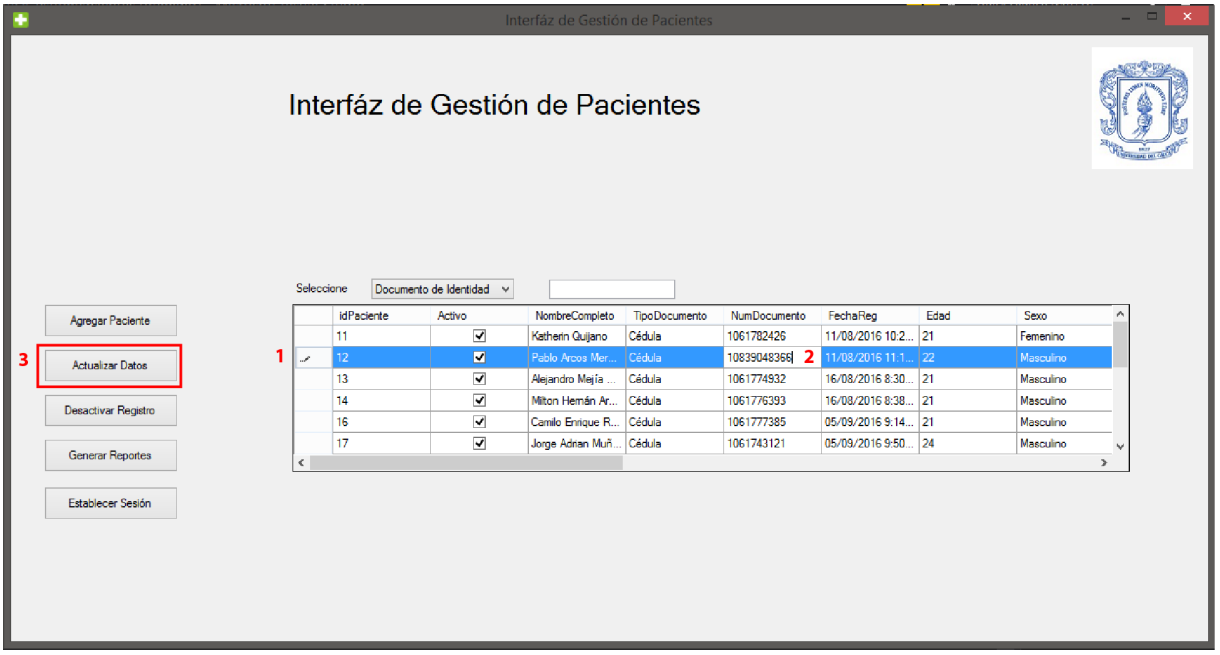

Figura C.3: Pasos para actualizar datos de un paciente

En la figura C.3 se puede ver gráficamente los pasos requeridos para ejecutar la actualización de los datos de un paciente. Es importante mencionar que solo se deben modificar los datos de un paciente al tiempo, ya que si se trata de modificar varios a la vez, no se guardaran los cambios en la base de datos.

C.1.3. Desactivar registro

Cuando sea necesario o se desee eliminar un paciente de la tabla de pacientes activos, se debe seleccionar el paciente a desactivar y posteriormente hacer click sobre el botón “Desactivar Registro”. Esta acción actualizará el estado del paciente seleccionado a *desactivado* lo cual significa que, éste no aparecerá en la tabla de pacientes activos de la aplicación. Sin embargo, los datos del paciente desactivado aún se encuentran en la base de datos y pueden ser recuperados cambiando el estado directamente desde la misma.

En la figura C.4 se muestra secuencialmente el proceso para desactivar un paciente, listando el proceso se vería como:

1. Seleccione el paciente que desea desactivar haciendo click en la primera columna de la tabla de la fila a la cual pertenece. Cuando ésta fila esté resaltada en color azul se puede proceder a desactivar el registro.
2. Dar click en el botón desactivar registro.
3. Se podrá apreciar que la información del paciente desactivado ya no será visible en la tabla informativa.

Figura C.4: Pasos para actualizar desactivar un paciente

C.1.4. Generar reportes

En la aplicación de gestión de pacientes, se tiene la posibilidad de generar reportes tanto globales como individuales. Para el caso de los reportes globales se muestran estadísticas concernientes al total de la población activa. Los datos mostrados incluyen la distribución de edades, distribución de genero, diagrama de pastel del porcentaje de los distintos tipos de ECV, etc.

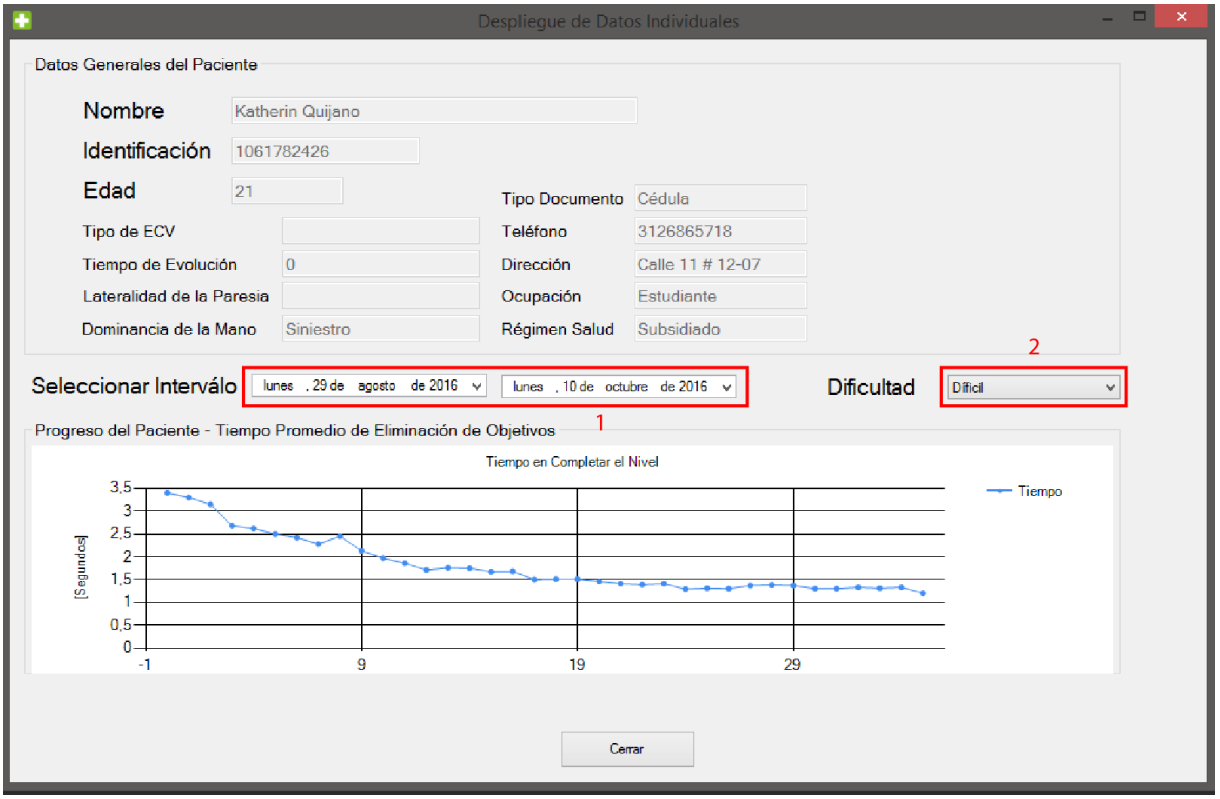

Figura C.5: Pasos para generar reportes individuales

La figura C.5 muestra el procedimiento a seguir para generar reportes individuales del progreso de un paciente. En primer lugar se debe seleccionar un paciente en la interfaz principal de gestión de pacientes (figura C.1) y a continuación, se debe presionar el botón "Generar Reportes". Una vez hecho ésto, aparecerá una nueva interfaz con la misma estructura que la mostrada en la figura C.5, en ésta se debe fijar el intervalo a graficar, el cual se selecciona usando los dos calendarios que se encuentran representados como botones en la aplicación. Una vez fijada la fecha, se debe seleccionar el nivel de dificultad de las sesiones que se requieren graficar. Estos niveles fueron fijados previamente por el terapeuta y se puede graficar cualquier nivel de dificultad realizada por el paciente. Una vez seleccionada la dificultad, en la parte inferior central

de la ventana se desplegará el gráfico de las sesiones realizadas por el paciente específico a lo largo del tiempo. Particularmente, cada punto desplegado indica una sesión en donde su valor en la ordenada corresponde al tiempo promedio en el que el paciente tarda en eliminar los objetivos y el valor en la abscisa es el número de la sesión realizada.

Figura C.6: Reporte global

Por otro lado, para generar reportes globales se debe hacer click en el botón “Generar Reportes” sin tener seleccionado ningún paciente o al seleccionar varios pacientes en la interfaz principal del gestor de pacientes. Cuando se presenta uno de estos casos el sistema despliega una ventana con el reporte de los datos globales de la población de pacientes, ver figura C.6.

C.1.5. Establecer sesión

El botón “Establecer Sesión” permite cargar a la base de datos el paciente que va a realizar la sesión de rehabilitación en el entorno virtual. Para asignar un paciente, simplemente debe seleccionar la fila correspondiente en la tabla de pacientes y presionar dicho botón, ver figura C.7. Para verificar que el paciente se haya establecido correctamente, asegúrese que aparezca el mensaje que se muestra en la figura C.7, este mensaje indica que la operación se realizó correctamente y se estableció el valor correspondiente en la base de datos.

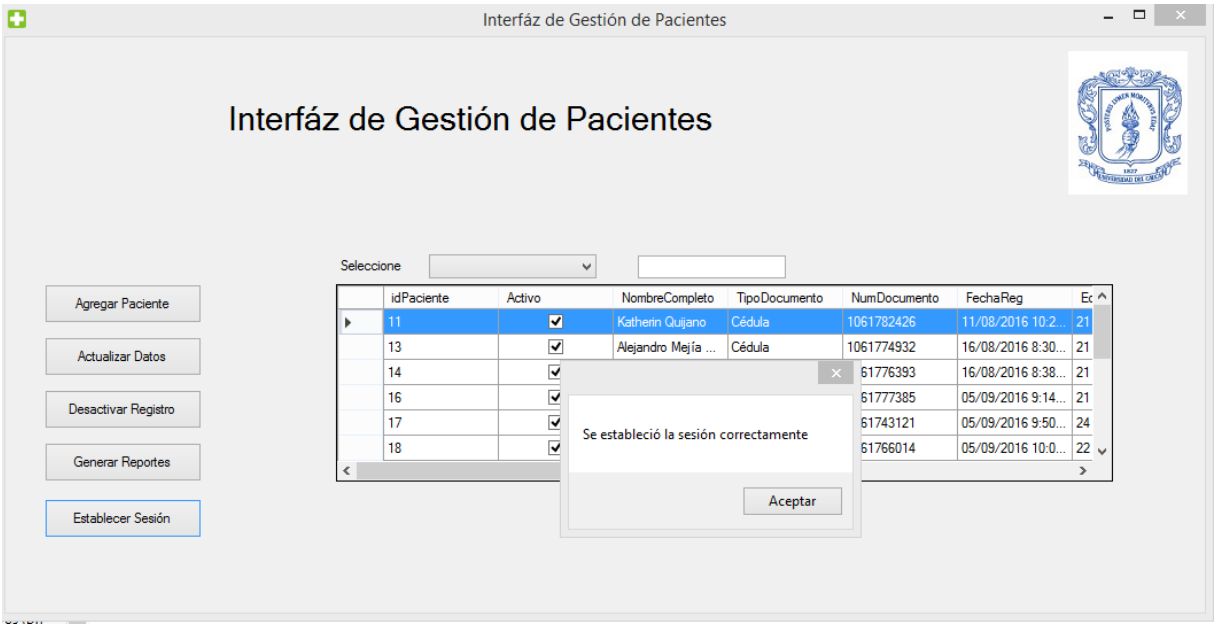

Figura C.7: Establecimiento de la sesión de rehabilitación

Posteriormente se ejecuta la aplicación de realidad virtual y el paciente seleccionado puede comenzar su sesión de rehabilitación. De esta forma se asocia el paciente seleccionado al entorno virtual y se evita tener que realizar configuraciones adicionales dentro de el entorno, facilitando así la usabilidad del mismo.

C.2. Aplicación de realidad virtual

A continuación se muestra una guía que explica en forma rápida, las funcionalidades más importantes del entorno virtual. En la figura C.8 se puede ver la ventana principal de la aplicación, la cual incluye los botones que ejecutan las distintas funciones dentro de la misma, entre éstas se incluye el botón de opciones, el botón de jugar y el botón del nivel tutorial. A continuación, se explicará como utilizar cada una de éstas funciones.

Figura C.8: Ventana principal de la aplicación

C.2.1. Configurar opciones

La configuración del juego es completamente personalizable, para ingresar a la ventana de configuración simplemente debe hacer click sobre el botón “Opciones” el cual desplegara inmediatamente la ventana mostrada en la figura C.9. En esta ventana se muestran los controles para cambiar la configuración del juego en forma intuitiva. Utilizando el *slider* que viene con la etiqueta “Tiempo de la Partida” se puede configurar la duración de cada partida de juego. Se recomienda un valor de 3 minutos para no generar cansancio en los pacientes.

Figura C.9: Ventana de opciones de la aplicación

Por otro lado, para cambiar el nivel de dificultad se debe hacer click en la lista desplegable indicada con la etiqueta “Nivel de Dificultad”, en esta lista se pueden ver los niveles disponibles y estos están descritos en la monografía del trabajo de grado. Otro aspecto importante de esta ventana de configuración, es la posibilidad de personalizar las variables que definen parámetros importantes dentro del juego. Por ejemplo variables hápticas como vibración, viscosidad y realimentación de fuerza y variables del escenario de juego como el tiempo de aparición de los objetivos y su tiempo de desaparición. Cabe aclarar, que se recomienda que ésta parte de configuración sea ejecutada principalmente por el fisioterapeuta a cargo del paciente a tratar, ya que éste posee los criterios suficientes para realizar dicha actividad.

C.2.2. Nivel de tutorial

La aplicación cuenta con un nivel de tutorial, éste le permite a los pacientes familiarizarse con el entorno virtual y adquirir una abstracción espacial que les permita desarrollar el ejercicio de rehabilitación planteado en las mejores condiciones y sin añadir retos cognitivos indeseados. La característica principal de éste nivel es que no se guardan datos en la base de datos y además, tampoco requiere una configuración del tiempo de la partida, lo cual permite jugar este nivel un tiempo definido por el terapeuta a cargo de la sesión de rehabilitación.

En la figura C.10 se muestra la ventana del escenario del tutorial. Este escenario corresponde al mismo escenario que se utiliza en los ejercicios de rehabilitación, lo cual ayuda a familiarizarse con el entorno virtual.

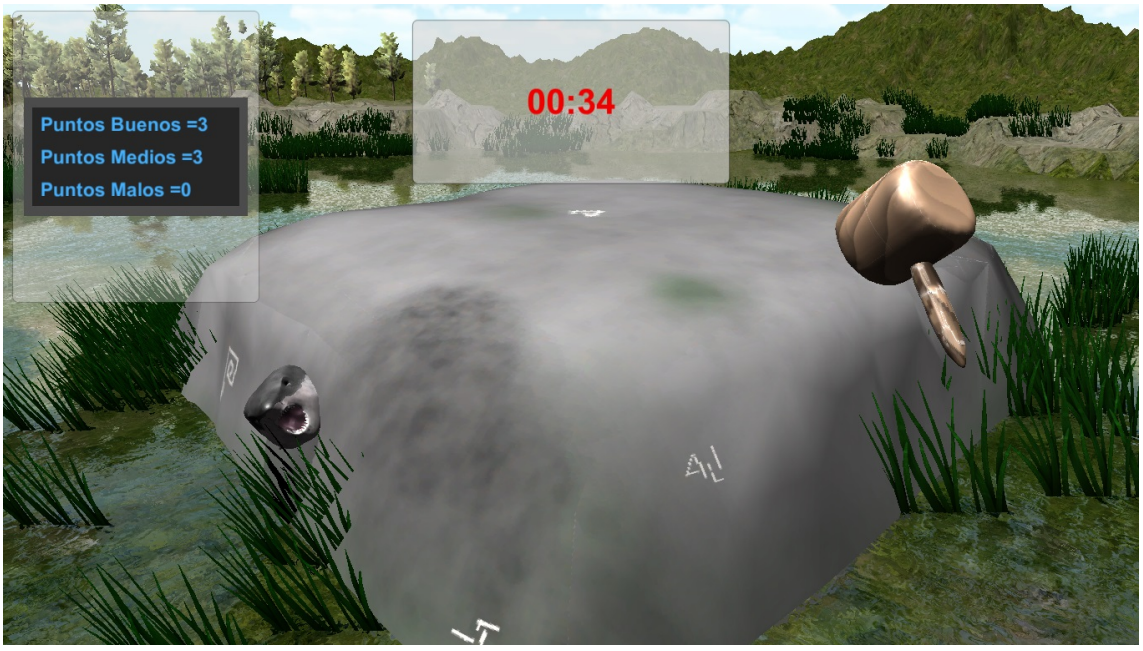

Figura C.10: Escenario del tutorial

APÉNDICE D

Guía para la configuración de Sqlite en Unity3D

Para el almacenamiento de los datos de los pacientes y las mediciones del desempeño de los mismos en el entorno virtual se requiere realizar la configuración de una base de datos *Sqlite*, tanto el archivo de la base de datos como el instalador de *Sqlite* son provistos dentro del directorio "Base de Datos". Para instalar *Sqlite* debe ser ejecutado el archivo "sqlite-netFx46-setup-bundle-x64-2015-1.0.98.0.exe", éste instalará las librerías necesarias para el funcionamiento de la base de datos en el entorno virtual y en la interfaz de gestión de pacientes. En la figura D.1 se pueden ver el archivo de instalación y el archivo de la base de datos llamado "DBPacientes.db" el cual contiene la estructura relacional de la base de datos.

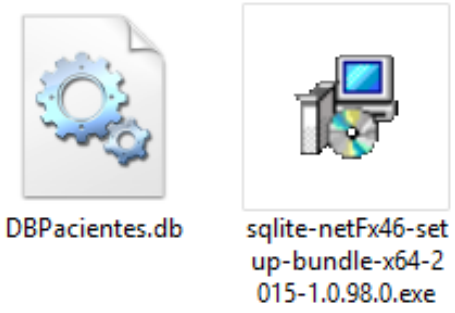

Figura D.1: Archivos de instalación de la base de datos

A continuación, se muestra una integración básica dentro del entorno de desarrollo Unity3D. Para realizar la integración de la base de datos *Sqlite*, se requieren los archivos mostrados en la figura D.2, estos se encuentran el el directorio **Base de Datos//Unity**. En otras palabras, para utilizar sentencias SQL dentro de Unity3D (necesarias para la consulta a la base de datos) se requiere importar dichos archivos dentro del proyecto a desarrollar (Unity3D).

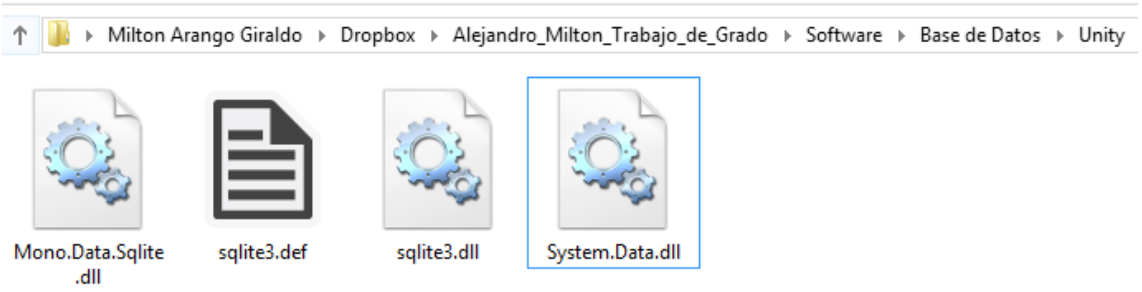

Figura D.2: Archivos de Sqlite para Unity3D

Para realizar la importación a Unity3D se requiere crear una carpeta llamada *Plugins*, en éste caso, Unity3D buscará los archivos adicionales de *Sqlite* dentro de esta carpeta en forma predeterminada. Para crear la carpeta, haga click derecho sobre la carpeta *Assets*, esto desplegará el menú mostrado en la figura D.3. Posteriormente seleccione la opción de carpeta (*Folder*) y nombrela como *Plugins*.

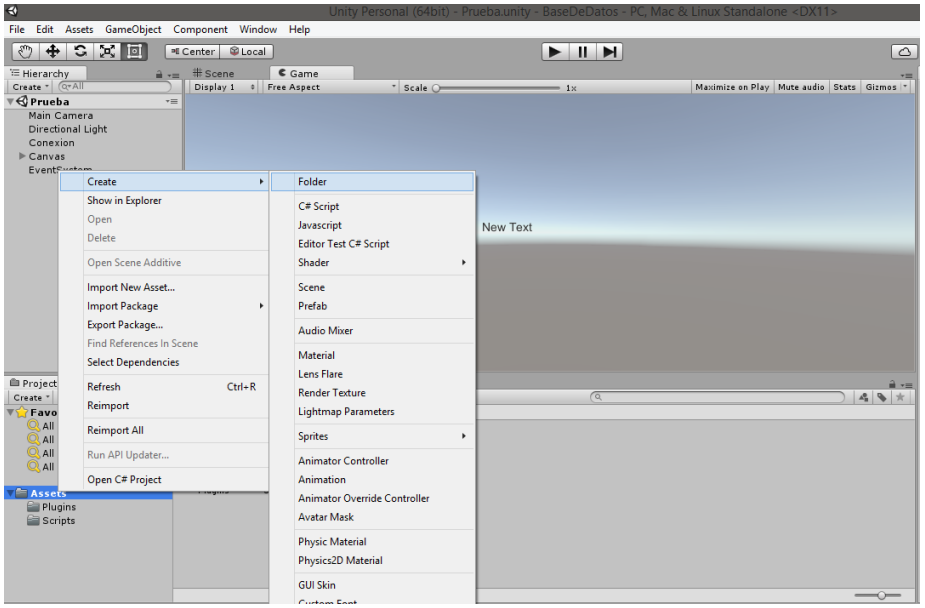

Figura D.3: Creación de la carpeta en Unity

Una vez realizados los pasos anteriores, es hora de importar los archivos a Unity3D. Para realizar esto, simplemente se deben arrastrar los archivos hasta la carpeta *Plugins* o copiar y pegar dentro de esta carpeta. De cualquiera de estos dos modos funciona. Para realizar una prueba con la base de datos se puede ver el código D.1 en el cual se muestra un *Script* básico de conexión a la base de datos.

Código D.1: Código para conectar la base de datos a Unity

```
1  using UnityEngine;
2  using System.Collections;
3  using Mono.Data.Sqlite;
4  using System;
5  using System.Data;
6
7  public class Connect : MonoBehaviour {
8
9          // Use this for initialization
10         void Start () {
11                 // Ubicacion base de datos
12                 string conn = "Data Source=" + Application.dataPath + "/DBP
                       acientes.db";
13                 IDbConnection dbconn;
14                 dbconn = (IDbConnection) new SqliteConnection(conn);
15                 dbconn.Open();
16                 IDbCommand dbcmd = dbconn.CreateCommand();
17                 string sqlQuery = "SELECT idPaciente FROM sesion_actual";
18                 dbcmd.CommandText = sqlQuery;
19                 IDataReader reader = dbcmd.ExecuteReader();
20                 int idPaciente = 0;
21
22                 while (reader.Read()){
23                         idPaciente = reader.GetInt32 (0);
24                         Debug.Log( "Id Paciente:" + idPaciente);
25                 }
26
27                 reader.Close();
28                 string Query = "SELECT * FROM Pacientes WHERE idPaciente =
                       " + idPaciente.ToString() + ";";
29                 dbcmd = dbconn.CreateCommand();
30                 dbcmd.CommandText = Query;
31                 reader = dbcmd.ExecuteReader();
32                 Patient paciente = new Patient();
```

```
33                  while (reader.Read()){
34                          paciente.FullName = reader.GetString (2);
35                          paciente.DocType = reader.GetString (3);
36                          paciente.IdNumber = reader.GetString (4);
37                          paciente.RegDate = reader.GetDateTime (5);
38                          paciente.Age = reader.GetInt16 (6);
39                          paciente.Sex = reader.GetString (7);
40                          paciente.CivState = reader.GetString (8);
41                          break;
42                  }
43                  Debug.Log( "Paciente:" + paciente.FullName);
44                  reader.Close();
45                  reader = null;
46                  dbcmd.Dispose();
47                  dbcmd = null;
48                  dbconn.Close();
49                  dbconn = null;
50          }
51
52          // Update is called once per frame
53          void Update () {
54          }
55  }
```

Bibliografía

[1] StarTech, "Startech 4 port pci 1394a firewire adapter card," 2002, [Online; accessed October 22, 2016]. [Online]. Available: https://images-na.ssl-images-amazon.com/images/I/91qpfcBmjmL._SL1500_.jpg

[2] *Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/UnityManual.html.

[3] *Descargando e Instalando Unity, Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/InstallingUnity.html.

[4] *Aprendiendo la Interfaz, Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/LearningtheInterface.html.

[5] M. Poyade, M. Kargas, and V. Portela, *Haptic Plug-in for Unity*, Digital Design Studio - The Glasgow School of Art, The Hub, Pacific Quay, G51 1EA, Glasgow, UK, June 2014.

[6] *Asset Packages (Paquetes de assets), Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/AssetPackages.html.

[7] *Importando desde la Asset Store, Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/AssetStore.html.

[8] *Etiquetas, Manual de Unity*, Unity Technologies, disponible en https://docs.unity3d.com/es/current/Manual/Tags.html.