2.6 CALIBRAR MICROSCOPIO

Dlg_Calib.h [Diseño]

Esta es la interfaz del segundo formulario secundario, con todos los elementos necesarios para caracterizar la configuración del microscopio que se va a usar para tomar las medidas y contar los anillos de un otolito. Y lo más importante, calcular la relación entre medidas en píxeles en el ordenador, con medidas reales sobre los transectos.

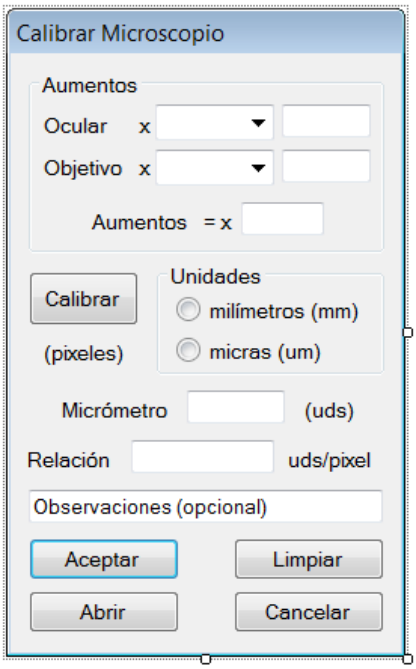

Figura 25.- Formulario de Calibración del Microscopio

Dlg_Calib.h

La diferencia con el formulario de Información General está en el botón Calibrar, el cual llama a funciones que se mostrarán más adelante. Por lo demás, tiene las mismas propiedades para el envío de datos y el control de fallos. El botón Abrir queda sin tratar pero también supondrá un punto clave para este formulario. El código se muestra a continuación.

```
1    #pragma once
2
3    #include <iostream>
4    #include <stdlib.h>
5    #include <string>
6    #include <vcclr.h>
7
...  ...
14
15   // para acortar codigo
16   #define DlgRes System::Windows::Forms::DialogResult
17
18   // declaraciones de funciones
19   double Conectar(void);
20   double CalibrarDesdeArchivo(char *fileAddress);
21
22   namespace OTOLIVE {
23
24      /// <summary>
25      /// Resumen de Dlg_Calib
26      ///
27      /// ADVERTENCIA: si cambia el nombre de esta clase, deberá cambiar la
28      ///          propiedad 'Nombre de archivos de recursos' de la herramienta de
     compilación de recursos administrados
29      ///          asociada con todos los archivos .resx de los que depende esta
     clase. De lo contrario,
30      ///          los diseñadores no podrán interactuar correctamente con los
31      ///          recursos adaptados asociados con este formulario.
32      /// </summary>
33      public ref class Dlg_Calib : public System::Windows::Forms::Form
34      {
35      public:
36         // definicion de variables para comunicacion con otros procesos, internos y
     externos
37         Dlg_Calib(void) :
38               condOcular(false)
39            , condObjetivo(false)
40            , condCalibrar(false)
41            , condMilimetros(false)
42            , condMicras(false)
43            , condMicrometro(false)
44            , condObservaciones(false)
45         {
46            InitializeComponent();
47            //
48            //TODO: agregar código de constructor aquí
49            //
50         }
51
52      protected:
53         /// <summary>
54         /// Limpiar los recursos que se estén utilizando.
55         /// </summary>
56         ~Dlg_Calib()
57         {
58            if (components)
59            {
60               delete components;
61            }
62         }
63
...  ...
98      private:
99         /// <summary>
100        /// Variable del diseñador requerida.
101        String ^orig;   // para guardar la ruta de la imagen
102        /// </summary>
103        System::ComponentModel::Container ^components;
104
...  ...
501  #pragma endregion
502
503        // variables de comprobacion de errores
504        bool condOcular;
505        bool condObjetivo;
506        bool condCalibrar;
507        double pixeles;
```

```
508         double micrometro;
509         bool condMilimetros;
510         bool condMicras;
511         bool condMicrometro;
512         double relacion;
513         bool condObservaciones;
514         // para usar #define DlgRes System::Windows::Forms::DialogResult y acortar
    codigo
515         DlgRes result;
516
517         // propiedades para el envio de datos
518         public: property String^ Ocular
519         {
520            String^ get()
521            {
522               return textBoxOcular->Text;
523            }
524         }
525
526         public: property String^ Objetivo
527         {
528            String^ get()
529            {
530               return textBoxObjetivo->Text;
531            }
532         }
533
534         public: property String^ Aumentos
535         {
536            String^ get()
537            {
538               return textBoxAumentos->Text;
539            }
540         }
541
542         public: property String^ Unidades
543         {
544            String^ get()
545            {
546               if (rdBtnMilimetros->Checked)
547                  return "Milímetros";
548               else if (rdBtnMicras->Checked)
549                  return "Micras";
550               else
551                  return "";
552            }
553         }
554
555         public: property String^ Micrometro
556         {
557            String^ get()
558            {
559               return textBoxMicrometro->Text;
560            }
561         }
562
563         public: property double Pixeles
564         {
565            double get()
566            {
567               return pixeles;
568            }
569         }
570
571         public: property String^ Relacion
572         {
573            String^ get()
574            {
575               return textBoxRelUdsPixel->Text;
576            }
577         }
578
579         public: property String^ Observaciones
580         {
581            String^ get()
582            {
583               return textBoxObservaciones->Text;
```

```
584             }
585         }
586
587         // comprobaciones al rellenar los campos
588         private: System::Void comboBoxOcular_KeyPress(System::Object^  sender,
    System::Windows::Forms::KeyPressEventArgs^  e)
589         {
590             // no permite introducir ningun valor
591             if (!Char::IsLetterOrDigit(e->KeyChar) || Char::IsLetterOrDigit(e-
    >KeyChar))
592                 e->Handled = true;
593         }
594
595         private: System::Void comboBoxOcular_SelectedIndexChanged(System::Object^
    sender, System::EventArgs^  e)
596         {
597             if (comboBoxOcular->SelectedIndex == 1)
598             {
599                 textBoxOcular->Enabled = true;
600                 textBoxOcular->Text = "";
601                 condOcular = false;
602             }
603             else
604             {
605                 textBoxOcular->Enabled = false;
606                 textBoxOcular->Text = comboBoxOcular->Text;
607                 condOcular = true;
608             }
609             if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
    condMicras) && condMicrometro)
610                 btnOK_Calib->DialogResult::set(DlgRes::OK);
611             btnLimpiar_Calib->Enabled = true;
612             btnOK_Calib->Enabled = true;
613         }
614
615         private: System::Void textBoxOcular_KeyPress(System::Object^  sender,
    System::Windows::Forms::KeyPressEventArgs^  e)
616         {
617             // solo permite una coma decimal
618             if (e->KeyChar == ',')
619             {
620                 if (textBoxOcular->Text->Contains(",") && !textBoxOcular->SelectedText-
    >Contains(","))
621                     e->Handled = true;
622             }
623             // acepta solo numeros, la coma decimal y la tecla de retroceso
624             else if (!Char::IsDigit(e->KeyChar) && e->KeyChar != 0x08)
625                 e->Handled = true;
626         }
627
628         private: System::Void textBoxOcular_TextChanged(System::Object^  sender,
    System::EventArgs^  e)
629         {
630             if (textBoxOcular->Text != "" && textBoxObjetivo->Text != "")
631                 textBoxAumentos->Text =
    System::Convert::ToString(System::Convert::ToDouble(textBoxOcular->Text) *
    System::Convert::ToDouble(textBoxObjetivo->Text));
632
633             condOcular = true;
634             if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
    condMicras) && condMicrometro)
635                 btnOK_Calib->DialogResult::set(DlgRes::OK);
636         }
637
638         private: System::Void comboBoxObjetivo_KeyPress(System::Object^  sender,
    System::Windows::Forms::KeyPressEventArgs^  e)
639         {
640             // no permite introducir ningun valor
641             if (!Char::IsLetterOrDigit(e->KeyChar) || Char::IsLetterOrDigit(e-
    >KeyChar))
642                 e->Handled = true;
643         }
644
645         private: System::Void comboBoxObjetivo_SelectedIndexChanged(System::Object^
    sender, System::EventArgs^  e)
646         {
647             if (comboBoxObjetivo->SelectedIndex == 5)
```

```
648           {
649              textBoxObjetivo->Enabled = true;
650              textBoxObjetivo->Text = "";
651              condObjetivo = false;
652           }
653           else
654           {
655              textBoxObjetivo->Enabled = false;
656              textBoxObjetivo->Text = comboBoxObjetivo->Text;
657              condObjetivo = true;
658           }
659           if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
    condMicras) && condMicrometro)
660              btnOK_Calib->DialogResult::set(DlgRes::OK);
661           btnLimpiar_Calib->Enabled = true;
662           btnOK_Calib->Enabled = true;
663       }
664
665       private: System::Void textBoxObjetivo_KeyPress(System::Object^  sender,
    System::Windows::Forms::KeyPressEventArgs^  e)
666       {
667          // solo permite una coma decimal
668          if (e->KeyChar == ',')
669          {
670             if (textBoxObjetivo->Text->Contains(",") && !textBoxObjetivo-
    >SelectedText->Contains(","))
671                e->Handled = true;
672          }
673          // acepta solo numeros, la coma decimal y la tecla de retroceso
674          else if (!Char::IsDigit(e->KeyChar) && e->KeyChar != 0x08)
675             e->Handled = true;
676       }
677
678       private: System::Void textBoxObjetivo_TextChanged(System::Object^  sender,
    System::EventArgs^  e)
679       {
680          if (textBoxOcular->Text != "" && textBoxObjetivo->Text != "")
681             textBoxAumentos->Text =
    System::Convert::ToString(System::Convert::ToDouble(textBoxOcular->Text) *
    System::Convert::ToDouble(textBoxObjetivo->Text));
682
683          condObjetivo = true;
684          if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
    condMicras) && condMicrometro)
685             btnOK_Calib->DialogResult::set(DlgRes::OK);
686       }
687
688       private: System::Void btnCalibrar_Click(System::Object^  sender,
    System::EventArgs^  e)
689       {
690          // llamamos a la funcion que conecta con la camara
691          pixeles = Conectar();
692
693          // si se ha realizado ua medida correctamente
694          if (pixeles > 0)
695             condCalibrar = true;   // validamos la condicion
696          // si no, se abre una imagen desde archivo
697          else if (MessageBox::Show("No hay cámara disponible\n¿Desea abrir imagen
    desde archivo?", "Información", MessageBoxButtons::OKCancel,
    MessageBoxIcon::Warning) == DlgRes::OK)
698          {
699             // guardamos la ruta en tipo String^
700             openFileDialog1->InitialDirectory = folderBrowserDialog1->SelectedPath;
701             openFileDialog1->FileName = nullptr;
702             openFileDialog1->ShowDialog();
703             orig = openFileDialog1->FileName;
704
705             // convertimos de String^ a char* (cadena de caracteres)
706             pin_ptr<const wchar_t> wch = PtrToStringChars(orig);
707             size_t origsize = wcslen(wch) + 1;
708             const size_t newsize = 200;
709             size_t convertedChars = 0;
710             char nstring[newsize];
711             wcstombs_s(&convertedChars, nstring, origsize, wch, _TRUNCATE);
712
713             // si se ha dado una ruta correcta
714             if (strcmp(nstring, "") != 0)
```

```
715             {
716                // llamada a la funcion que muestra y mide un segmento en una imagen
717                pixeles = CalibrarDesdeArchivo(nstring);
718                condCalibrar = true;
719             }
720          }
721
722          String^ numPx;
723          numPx = System::Convert::ToString((int)pixeles);
724          numPx += " px";
725          labelPixeles->Text = numPx;
726          groupBox2->Enabled = true;
727          if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
condMicras) && condMicrometro)
728             btnOK_Calib->DialogResult::set(DlgRes::OK);
729          btnLimpiar_Calib->Enabled = true;
730          btnOK_Calib->Enabled = true;
731       }
732
733       private: System::Void rdBtnMilimetros_CheckedChanged(System::Object^  sender,
System::EventArgs^  e)
734       {
735          labelMicrometro->Text = "(mm)";
736          labelRelacion->Text = "mm/pixel";
737          textBoxMicrometro->Enabled = true;
738          condMilimetros = true;
739          condMicras = false;
740          if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
condMicras) && condMicrometro)
741             btnOK_Calib->DialogResult::set(DlgRes::OK);
742       }
743
744       private: System::Void rdBtnMicras_CheckedChanged(System::Object^  sender,
System::EventArgs^  e)
745       {
746          labelMicrometro->Text = "(um)";
747          labelRelacion->Text = "um/pixel";
748          textBoxMicrometro->Enabled = true;
749          condMicras = true;
750          condMilimetros = false;
751          if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
condMicras) && condMicrometro)
752             btnOK_Calib->DialogResult::set(DlgRes::OK);
753       }
754
755       private: System::Void textBoxMicrometro_KeyPress(System::Object^  sender,
System::Windows::Forms::KeyPressEventArgs^  e)
756       {
757          // solo permite una coma decimal
758          if (e->KeyChar == ',')
759          {
760             if (textBoxObjetivo->Text->Contains(",") && !textBoxObjetivo-
>SelectedText->Contains(","))
761                e->Handled = true;
762          }
763          // acepta solo numeros, la coma decimal y la tecla de retroceso
764          else if (!Char::IsDigit(e->KeyChar) && e->KeyChar != 0x08)
765             e->Handled = true;
766       }
767
768       private: System::Void textBoxMicrometro_TextChanged(System::Object^  sender,
System::EventArgs^  e)
769       {
770          if (textBoxMicrometro->Text == "")
771             relacion = 0.0;
772          else
773          {
774             micrometro = System::Convert::ToDouble(textBoxMicrometro->Text);
775             relacion = micrometro / pixeles;
776             textBoxRelUdsPixel->Text = System::Convert::ToString(relacion);
777          }
778          condMicrometro = true;
779          if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
condMicras) && condMicrometro)
780             btnOK_Calib->DialogResult::set(DlgRes::OK);
781       }
782
```

```
783         private: System::Void textBoxObservaciones_TextChanged(System::Object^
    sender, System::EventArgs^  e)
784         {
785            condObservaciones = true;
786            if (condOcular && condObjetivo && condCalibrar && (condMilimetros ||
    condMicras) && condMicrometro)
787               btnOK_Calib->DialogResult::set(DlgRes::OK);
788            btnLimpiar_Calib->Enabled = true;
789            btnOK_Calib->Enabled = true;
790         }
791
792         private: System::Void textBoxObservaciones_MouseDown(System::Object^  sender,
    System::Windows::Forms::MouseEventArgs^  e)
793         {
794            // quitamos el texto por defecto para poder escribir
795            textBoxObservaciones->Text = "";
796         }
797
798         // comprobacion final
799         private: System::Void btnOK_Calib_Click(System::Object^  sender,
    System::EventArgs^  e)
800         {
801            // cadena donde registramos los campos que falten por introducir
802            String^ errores = "";
803
804            if (!condOcular)
805               errores += "Ocular\n";
806            if (!condObjetivo)
807               errores += "Objetivo\n";
808            if (!condCalibrar)
809               errores += "Calibrar\n";
810            if (!condMilimetros && !condMicras)
811               errores += "Unidades\n";
812            if (!condMicrometro)
813               errores += "Micrómetro\n";
814            if (!condObservaciones)
815               errores += "(opcional: Observaciones)";
816
817            // si exinten campos sin introducir
818            if ((errores != "") && (errores != "(opcional: Observaciones)"))
819               MessageBox::Show(errores, "Faltan datos por introducir",
    MessageBoxButtons::OK, MessageBoxIcon::Warning);
820            else   // si todo es correcto
821            {
822               MessageBox::Show("Enviando campos a la base de datos", "Información",
    MessageBoxButtons::OK, MessageBoxIcon::Information);
823               // para acceder despues sin fallos
824               btnOK_Calib->Enabled = false;
825            }
826            // para volver a empezar
827            btnOK_Calib->DialogResult = DlgRes::None;
828         }
829
830         private: System::Void btnAbrir_Calib_Click(System::Object^  sender,
    System::EventArgs^  e)
831         {
832            // para abrir una imagen desde archivo
833         }
834
835         private: System::Void btnLimpiar_Calib_Click(System::Object^  sender,
    System::EventArgs^  e)
836         {
837            result = MessageBox::Show("¿Seguro que desea limpiar\nlos campos del
    formulario?", "Limpiar Base de Datos", MessageBoxButtons::OKCancel,
    MessageBoxIcon::Warning);
838            if (result == DlgRes::OK)
839            {
840               // se vacian los campos
841               comboBoxOcular->Text = "";
842               textBoxOcular->Text = "";
843               comboBoxObjetivo->Text = "";
844               textBoxObjetivo->Text = "";
845               textBoxAumentos->Text = "";
846               labelPixeles->Text = "(pixeles)";
847               rdBtnMilimetros->Checked = false;
848               rdBtnMicras->Checked = false;
849               groupBox2->Enabled = false;
```

```
850             textBoxMicrometro->Text = "";
851             textBoxMicrometro->Enabled = false;
852             textBoxRelUdsPixel->Text = "";
853             textBoxObservaciones->Text = "Observaciones (opcional)";
854
855             // se resetean las condiciones de comprobacion de errores
856             condOcular = false;
857             condObjetivo = false;
858             condCalibrar = false;
859             condMilimetros = false;
860             condMicras = false;
861             condMicrometro = false;
862             condObservaciones = false;
863
864             // se bloquea el boton hasta que vuelva a haber algo que limpiar
865             btnLimpiar_Calib->Enabled = false;
866             // para volver a empezar
867             btnOK_Calib->DialogResult = DlgRes::None;
868             btnOK_Calib->Enabled = true;
869
870             btnLimpiar_Calib->DialogResult = DlgRes::Yes;
871             result = MessageBox::Show("¿Desea salir del diálogo\nCalibrar
    Microscopio?", "Salir", MessageBoxButtons::OKCancel, MessageBoxIcon::Warning);
872             if (result == DlgRes::OK)
873             {
874                 // para limpiar campos de la base de datos
875                 Dlg_Calib::DialogResult = DlgRes::Yes;
876                 // se cierra el formulario
877                 Dlg_Calib::Close();
878             }
879         }
880     }
881 };
882 }
```

Los primeros puntos a considerar comienzan en la línea 3 con unas cabeceras que servirán para convertir una variable de tipo String^ a char*, y en las líneas 19 y 20, en las que se declaran las llamadas a funciones externas que realizarán tareas concretas de conexión con la cámara y de tratamiento de imágenes (estos archivos se estudiarán a fondo en el apartado Otras Funciones).También es interesante fijarse en la línea 101, lugar en el que se define la variable de tipo String^ que será convertida a char*.

Como ya se comentó en el apartado de Información General, en el formulario que nos encontramos estudiando ahora hay propiedades públicas de tipo double, no sólo String^, ya que se desea no sólo almacenar valores numéricos, si no también trabajar con ellos, hacer cálculos, y por esta razón conviene que no sean cadenas de caracteres.

En líneas de código como la 631 se realizan varias conversiones de tipo anidadas, ya que por ejemplo se quiere mostrar un valor numérico en un cuadro de texto del formulario, que consiste en el resultado de la multiplicación de otros datos, que a su vez se presentan en tipo cadena de caracteres. Entonces se convierten primero los datos a double, se multiplican, y este valor final vuelve a convertirse a cadena para poder mostrarlos fácilmente mediante el comando Text.

El evento más importante, es el que se encarga de dar respuesta a pulsar el botón Calibrar. Desde aquí, dependiendo de si la cámara está conectada o no, se llama respectivamente a las funciones Conectar y AbrirDesdeArchivo.

El botón Abrir queda reflejado pero inactivo, para que en futuros avances desempeñe una importante función de ahorro de tiempo a los investigadores, ya que se podrán cargar configuraciones del microscopio sin tener que rellenar todos los datos, si no valiéndose de otros documentos de la base de datos que posean características similares.