1dd_BladeEditor.cpp

```cpp
int T1dBladeEditorDlg::initializePage()
{
  // W01
  QVBoxLayout* grid = new QVBoxLayout(this);
  // W02
  w_PropertyHolderWidget* holder = new w_PropertyHolderWidget(this);

  // W03
  // 1.
  holder_3DViewWidget = holder->getHolder(0, 0, 2, 1, tr("3D"));
  _3DViewWidget = vis_Widget::newWidget("vis_WidgetVtk", holder_3DViewWidget);
  holder_3DViewWidget->placeWidget(_3DViewWidget);

  // 2
  holder_MerdionalCurvesWidget = holder->getHolder(0, 1, 1, 1, tr("Merdional"));

  _MerdionalCurvesWidget = new T1dBladeMeridionalCurveWidget(this);
  holder_MerdionalCurvesWidget->placeWidget(_MerdionalCurvesWidget);
  // W04
  grid->addWidget(holder);

  QPushButton* applyConfig = new QPushButton(tr("Config"));
  connect(applyConfig, SIGNAL(clicked()), this, SLOT(onConfigButtonPressed()));
  // W05
  QPushButton* applyShowThroat = new QPushButton(tr("Show Throat"));
  connect(applyShowThroat, SIGNAL(clicked()), this, SLOT(onShowThroatButton()));
  // W06
  buttonsLayout->addWidget(applyShowThroat);
  buttonsLayout->addStretch(1);

  buttonsLayout->addWidget(applyButton);
  buttonsLayout->addWidget(okButton);
  buttonsLayout->addWidget(cancelButton);
  // W07
  grid->addLayout(buttonsLayout);
  // W08
  setLayout(grid);
  setFocusPolicy(Qt::StrongFocus);
```

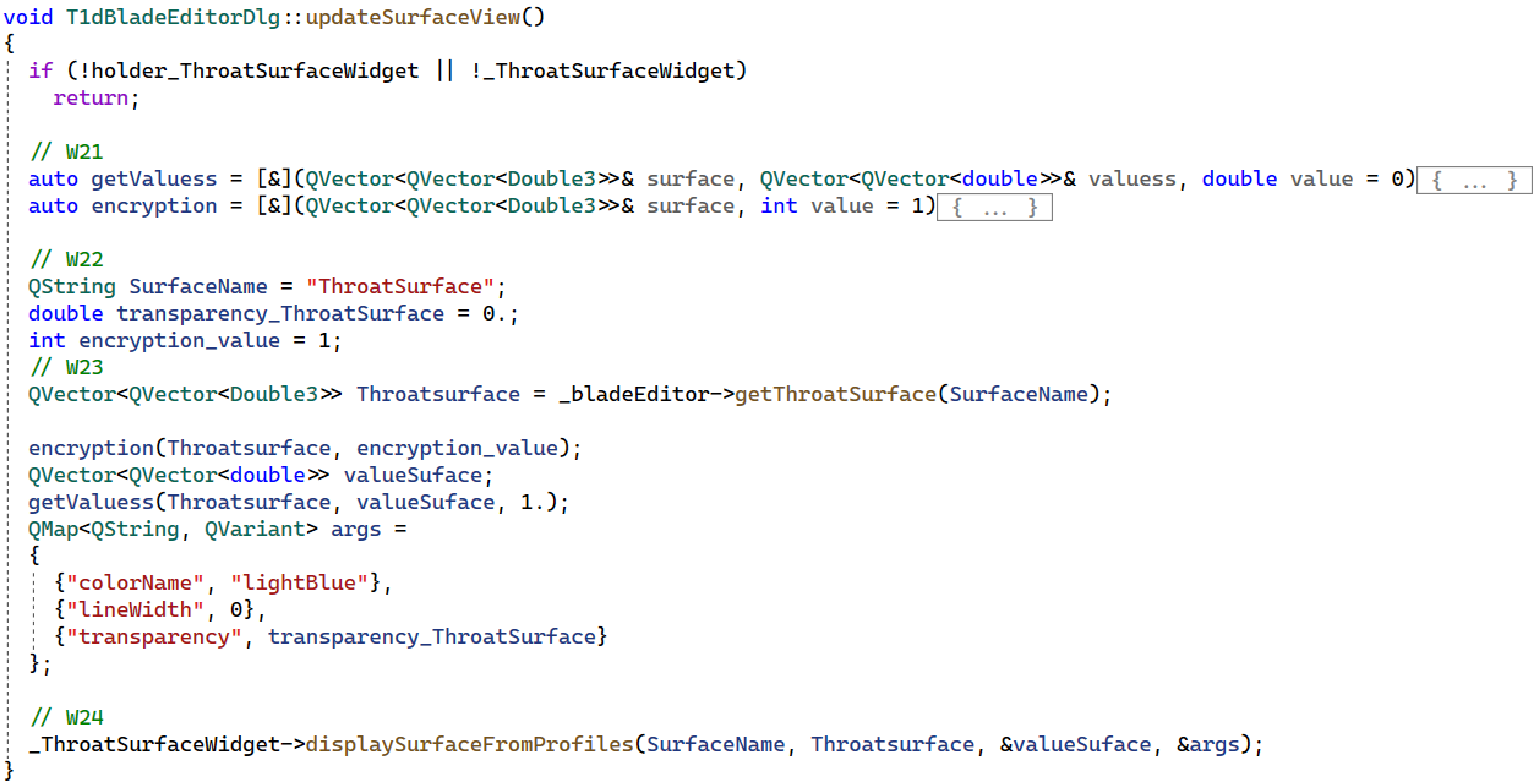
1d_BladeEditor.cpp

```cpp
REGISTER_OBJECT_CLASS(T1dBladeEditor, "Blade editor", TObject);
T1dBladeEditor::T1dBladeEditor(QString object_n, TObject* iparent) :
  TObject(object_n, iparent)
{
  INIT_OBJECT;
  DEFINE_SCALAR_INIT(double, o1tip, 0, 0, NULL, TUnit::length);
  DEFINE_SCALAR_INIT(double, o1mean, 0, 0, NULL, TUnit::length);
  DEFINE_SCALAR_INIT(double, o1hub, 0, 0, NULL, TUnit::length);
  DEFINE_SCALAR_INIT(double, ThroatArea, 0, 0, NULL, TUnit::area);
  _pVaned = nullptr;
}
```

1dd_BladeEditor.cpp

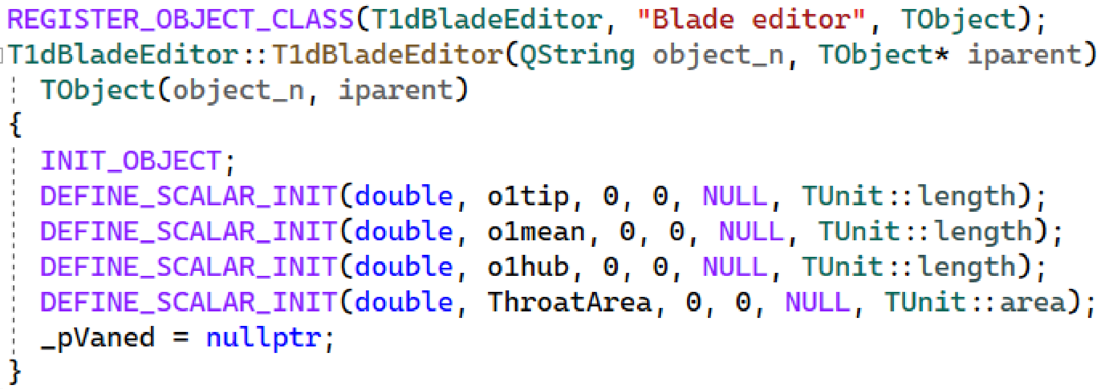
```
// written by Feihong
void T1dBladeEditorDlg::onShowThroatButton()
{
  // W11
  w_QDialog dlg(core_Application::core());
  dlg.setWindowTitle(w_QDialog::tr("Show Throat"));
  // W12
  QVBoxLayout* grid = new QVBoxLayout(&dlg);
  // W13
  w_PropertyHolderWidget* holder = new w_PropertyHolderWidget();

  // W14-1 vtk-3Dshow
  // 1a. w_PropertyHolderWidget* holder_ThroatSurfaceWidget;
  holder_ThroatSurfaceWidget = holder->getHolder(0, 0, 1, 2, tr("Throat surface"));
  // 1b. vis_Widget* _ThroatSurfaceWidget;
  _ThroatSurfaceWidget = vis_Widget::newWidget("vis_WidgetVtk", holder_ThroatSurfaceWidget);
  updateSurfaceView();
  // 1c.
  holder_ThroatSurfaceWidget->placeWidget(_ThroatSurfaceWidget);

  // W14-2 PropertyList
  // 2a. w_PropertyHolderWidget* holder_ThroatPropertyWidget;
  holder_ThroatSurfaceWidget = holder->getHolder(0, 2, 1, 2, tr("Throat propertylist"));
  // 2b. TPropertyInputWidget* _ThroatPropertyWidget;
  _ThroatPropertyWidget = new TPropertyInputWidget(holder_ThroatSurfaceWidget);
  // 2c.
  QVector<property_t*> properties = QVector<property_t*>()
    << _bladeEditor->property("o1tip") << _bladeEditor->property("o1mean")
    << _bladeEditor->property("o1hub") << _bladeEditor->property("ThroatArea");
  // 2d.
  _ThroatPropertyWidget->setProperties(properties, true);
  // 2e.
  holder_ThroatSurfaceWidget->placeWidget(_ThroatPropertyWidget);

  // W15
  grid->addWidget(holder);
  // W16
  setLayout(grid);
  setFocusPolicy(Qt::StrongFocus);

  if (dlg.exec() == w_QDialog::Accepted) {}
}
```

1dd_BladeEditor.cpp



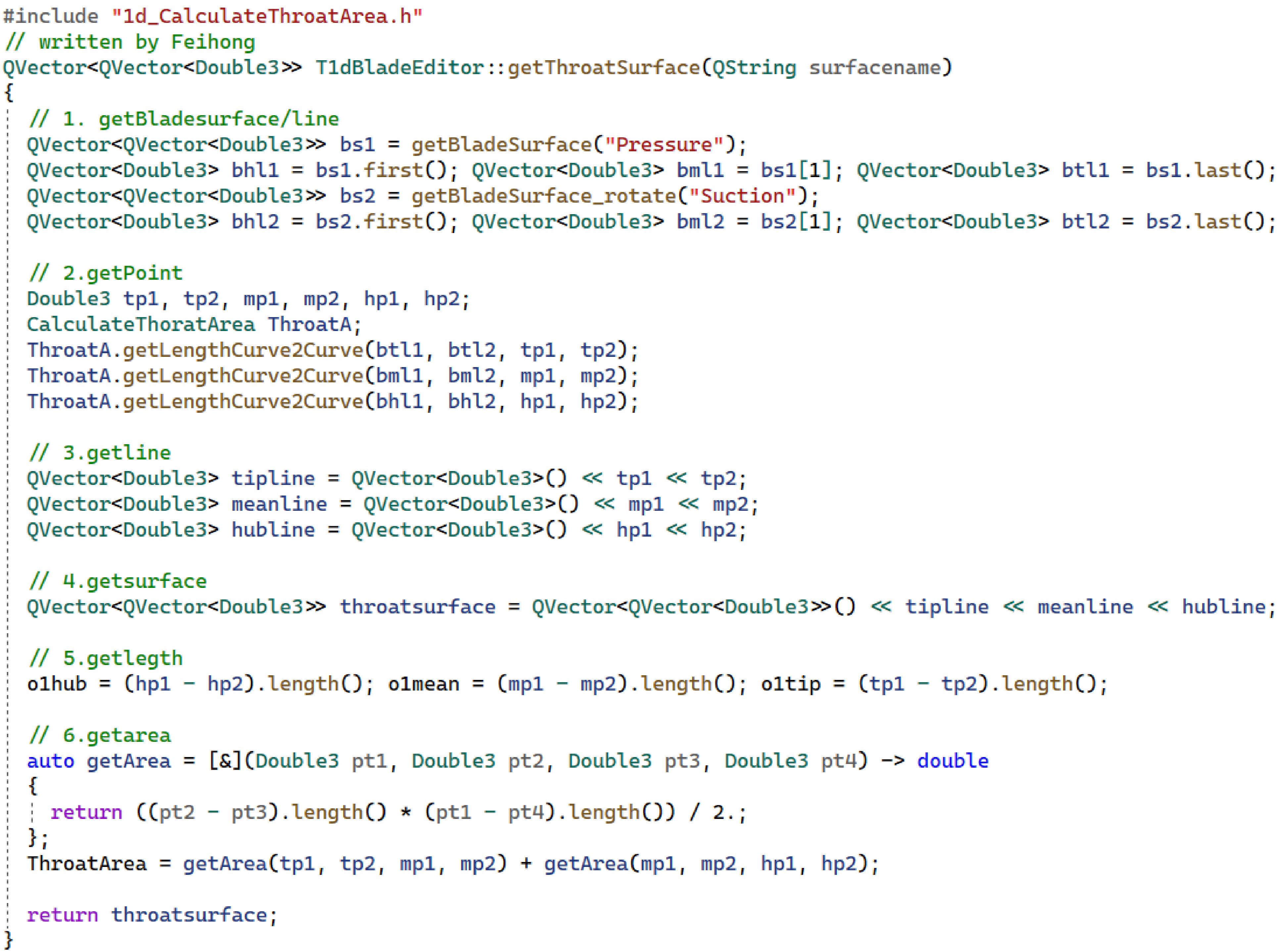
```cpp
void T1dBladeEditorDlg::updateSurfaceView()
{
    if (!holder_ThroatSurfaceWidget || !_ThroatSurfaceWidget)
        return;

    // W21
    auto getValuess = [&](QVector<QVector<Double3>>& surface, QVector<QVector<double>>& valuess, double value = 0){ ... }
    auto encryption = [&](QVector<QVector<Double3>>& surface, int value = 1){ ... }

    // W22
    QString SurfaceName = "ThroatSurface";
    double transparency_ThroatSurface = 0.;
    int encryption_value = 1;
    // W23
    QVector<QVector<Double3>> Throatsurface = _bladeEditor->getThroatSurface(SurfaceName);

    encryption(Throatsurface, encryption_value);
    QVector<QVector<double>> valueSuface;
    getValuess(Throatsurface, valueSuface, 1.);
    QMap<QString, QVariant> args =
    {
        {"colorName", "lightBlue"},
        {"lineWidth", 0},
        {"transparency", transparency_ThroatSurface}
    };

    // W24
    _ThroatSurfaceWidget->displaySurfaceFromProfiles(SurfaceName, Throatsurface, &valueSuface, &args);
}
```

1d_BladeEditor.cpp



```cpp
REGISTER_OBJECT_CLASS(T1dBladeEditor, "Blade editor", TObject);
T1dBladeEditor::T1dBladeEditor(QString object_n, TObject* iparent) :
    TObject(object_n, iparent)
{
    INIT_OBJECT;
    DEFINE_SCALAR_INIT(double, o1tip, 0, 0, NULL, TUnit::length);
    DEFINE_SCALAR_INIT(double, o1mean, 0, 0, NULL, TUnit::length);
    DEFINE_SCALAR_INIT(double, o1hub, 0, 0, NULL, TUnit::length);
    DEFINE_SCALAR_INIT(double, ThroatArea, 0, 0, NULL, TUnit::area);
    _pVaned = nullptr;
}
```

1d_BladeEditor.cpp



```cpp
#include "1d_CalculateThroatArea.h"
// written by Feihong
QVector<QVector<Double3>> T1dBladeEditor::getThroatSurface(QString surfacename)
{
    // 1. getBladesurface/line
    QVector<QVector<Double3>> bs1 = getBladeSurface("Pressure");
    QVector<Double3> bhl1 = bs1.first(); QVector<Double3> bml1 = bs1[1]; QVector<Double3> btl1 = bs1.last();
    QVector<QVector<Double3>> bs2 = getBladeSurface_rotate("Suction");
    QVector<Double3> bhl2 = bs2.first(); QVector<Double3> bml2 = bs2[1]; QVector<Double3> btl2 = bs2.last();

    // 2.getPoint
    Double3 tp1, tp2, mp1, mp2, hp1, hp2;
    CalculateThoratArea ThroatA;
    ThroatA.getLengthCurve2Curve(btl1, btl2, tp1, tp2);
    ThroatA.getLengthCurve2Curve(bml1, bml2, mp1, mp2);
    ThroatA.getLengthCurve2Curve(bhl1, bhl2, hp1, hp2);

    // 3.getline
    QVector<Double3> tipline = QVector<Double3>() << tp1 << tp2;
    QVector<Double3> meanline = QVector<Double3>() << mp1 << mp2;
    QVector<Double3> hubline = QVector<Double3>() << hp1 << hp2;

    // 4.getsurface
    QVector<QVector<Double3>> throatsurface = QVector<QVector<Double3>>() << tipline << meanline << hubline;

    // 5.getlegth
    o1hub = (hp1 - hp2).length(); o1mean = (mp1 - mp2).length(); o1tip = (tp1 - tp2).length();

    // 6.getarea
    auto getArea = [&](Double3 pt1, Double3 pt2, Double3 pt3, Double3 pt4) -> double
    {
        return ((pt2 - pt3).length() * (pt1 - pt4).length()) / 2.;
    };
    ThroatArea = getArea(tp1, tp2, mp1, mp2) + getArea(mp1, mp2, hp1, hp2);

    return throatsurface;
}
```

1dd_BladeEditor.cpp

```cpp
void T1dBladeEditorDlg::update3DView()
{
  if (!holder_bladeThicknessCurvesWidget || !_3DViewWidget)
    return;

  auto getValuess = [&](QVector<QVector<Double3>>& surface, QVector<QVector<double>>& valuess, double value = 0)
  {
    for (int i = 0; i < surface.size(); i++)
    {
      QVector<double> values;
      for (int j = 0; j < surface[i].size(); j++)
      {
        values.push_back(value);
      }
      valuess.push_back(values);
    }
  };

  auto encryption = [&](QVector<QVector<Double3>>& surface, int value = 1)
  {
    if (value < 1)
      return;

    double dt = 1. / (value+1.);

    for (int i = 0; i < surface.size(); i++)
    {
      int size = surface[i].size();

      for (int j = size - 1; j > 0; j--)
      {
        Double3 pt_start = surface[i][j];
        Double3 pt_end = surface[i][j - 1];

        for (int k = 0; k < value; k++)
        {
          double t = (k + 1) * dt;
          Double3 pt = (1-t) * pt_start +t* pt_end;
          surface[i].insert(j, pt);
        }
      }
    }
  };
```

```cpp
int errorCode = _bladeEditor->updateProfilesGenerator();
if (errorCode != 0)
  return;

// Blade 1
QStringList surfaceList = QStringList() << "Camber" << "LE" << "Pressure" << "TE" << "Suction";
QVector<double> transparency_bladeSurface = QVector<double>() <<  7. << 5. << 5. << 5. << 5.;
QVector<int> encryption_value = QVector<int>() << 1 << 1 << 1 << 1 << 1;
for (int i = 0; i< surfaceList.size(); i++)
{
  QVector<QVector<Double3>> surface = _bladeEditor->getBladeSurface(surfaceList[i]);

  encryption(surface, encryption_value[i]);

  QVector<QVector<double>> valueSuface;

  getValuess(surface, valueSuface,1.);

  QMap<QString, QVariant> args =
  {
    {"colorName", "lightBlue"},
    {"lineWidth", 0},
    {"transparency", transparency_bladeSurface[i]}
  };

  _3DViewWidget->displaySurfaceFromProfiles(surfaceList[i], surface, &valueSuface, &args);
}

// Blade 2
{
  QStringList surfaceList = QStringList() << "Camber1" << "LE1" << "Pressure1" << "TE1" << "Suction1";
  QVector<double> transparency_bladeSurface = QVector<double>() << 7. << 5. << 5. << 5. << 5.;
  QVector<int> encryption_value = QVector<int>() << 1 << 1 << 1 << 1 << 1;
  for (int i = 0; i < surfaceList.size(); i++)
  {
    QVector<QVector<Double3>> surface = _bladeEditor->getBladeSurface_rotate(surfaceList[i]);

    encryption(surface, encryption_value[i]);

    QVector<QVector<double>> valueSuface;

    getValuess(surface, valueSuface, 1.);
```

1dd_BladeEditor.cpp

```cpp
    // revolutionSurface
    QVector<double> spans = { 0., 100. }; // _bladeEditor->getSpans();
    for (int i = 0; i < spans.size(); i++)
    {
        QVector<QVector<Double3>> RevolutionSurface = _bladeEditor->getRevolutionSurface(spans[i])
        QVector<QVector<double>> valueSuface;
        getValuess(RevolutionSurface, valueSuface);

        double transparency = 0.;
        if (i == spans.size() - 1)
            transparency = 8.;
        QMap<QString, QVariant> args =
        {
            {"colorName", "lightBlue"},
            {"lineWidth", 0},
            {"transparency", transparency}
        };

        QString name = "RevolutionSurface" + _bladeEditor->getSpanString(spans[i]);
        _3DViewWidget->displaySurfaceFromProfiles(name, RevolutionSurface, &valueSuface, &args);
    }

    // ThroatSurface
    QString SurfaceName = "ThroatSurface";
    double transparency_ThroatSurface = 0.;
    // W23
    QVector<QVector<Double3>> Throatsurface = _bladeEditor->getThroatSurface(SurfaceName);

    encryption(Throatsurface, 5);
    QVector<QVector<double>> valueSuface;
    getValuess(Throatsurface, valueSuface, 1.);
    QMap<QString, QVariant> args =
    {
        {"colorName", "lightBlue"},
        {"lineWidth", 0},
        {"transparency", transparency_ThroatSurface}
    };

    // W24
    _3DViewWidget->displaySurfaceFromProfiles(SurfaceName, Throatsurface, &valueSuface, &args);
}
```

```cpp
void T1dBladeEditorDlg::onConfigButtonPressed()
{
    w_PropertyHolderDialog dlg(core_Application::core());
    dlg.setWindowTitle(w_PropertyHolderDialog::tr("Config"));

    w_PropertyHolderWidget* holder = dlg.getHolder();
    w_PropertyHolderWidget* holder1 = holder->addHolder();

    QStringList _allBladeType = QStringList() << tr("Blade angle") << tr("Conformal mapping");
    _bladeEditorType = _allBladeType[_bladeEditor->_bladeEditorType];

    w_Property* wType = nullptr;
    if (wType = w_Property::getPropertyWidget(&_bladeEditorType, QObject::tr("Blade editor type: "),
        holder1, &_allBladeType, true, false, w_Property::HLayout))
    {
        wType->setAutoSave();
        holder1->placeWidget(wType, 0, 0, 1, 1);
        connect(wType, SIGNAL(valueChanged()), this, SLOT(bladeEditorTypeChanged()));
    }
```