```cpp
#include "StdAfx.h"
#include "MenuBase.h"

extern int LibMessage(LPCTSTR str, int flags = MB_OK);

void MenuBase::initCoordinates(size_t id, size_t id_x, size_t id_y, size_t id_z) {
    BSTR name = _T("Координаты");
    this->coordinates = this->collection->Add(ControlTypeEnum::ksControlPoint3D);
    this->coordinates->Name = name;
    this->coordinates->Hint = name;
    this->coordinates->Tips = name;
    this->coordinates->Id = id;
    IPropertyEditPtr X = this->coordinates->GetCoordinate(0);
    IPropertyEditPtr Y = this->coordinates->GetCoordinate(1);
    IPropertyEditPtr Z = this->coordinates->GetCoordinate(2);

    X->Id = id_x;
    X->Value = this->getModel()->coordinates.X;
    X->CheckState = CheckStateEnum::ksCheckFixed;

    Y->Id = id_y;
    Y->Value = this->getModel()->coordinates.Y;
    Y->CheckState = CheckStateEnum::ksCheckFixed;

    Z->Id = id_z;
    Z->Value = this->getModel()->coordinates.Z;
    Z->CheckState = CheckStateEnum::ksCheckFixed;
}

void MenuBase::initSurfaceType(size_t id) {
    BSTR name = _T("Тип основания");
    this->surfaceType = this->collection->Add(ControlTypeEnum::ksControlListStr);
    this->surfaceType->Name = name;
    this->surfaceType->Hint = name;
    this->surfaceType->Tips = name;
    this->surfaceType->Id = id;
    this->surfaceType->ReadOnly = true;
    this->surfaceType->Add(_T("Круг"));
    this->surfaceType->Add(_T("Треугольник"));
    this->surfaceType->Add(_T("Квадрат"));
    this->surfaceType->Add(_T("Пятиугольник"));
    this->surfaceType->Add(_T("Шестиугольник"));
    this->surfaceType->Add(_T("Тригон"));
    this->surfaceType->Add(_T("Ромб 35"));
    this->surfaceType->Add(_T("Ромб 55"));
    this->surfaceType->Add(_T("Ромб 80"));
    this->surfaceType->SetCurrentByIndex(this->getModel()->surfaceType);
}

void MenuBase::initSizeType(size_t id) {
    BSTR name = _T("Тип размера");
    this->sizeType = this->collection->Add(ControlTypeEnum::ksControlListStr);
    this->sizeType->Name = name;
    this->sizeType->Hint = name;
    this->sizeType->Tips = name;
    this->sizeType->Id = id;
    this->sizeType->ReadOnly = true;
    this->sizeType->Add(_T("По радиусу описанной окружности"));
    this->sizeType->Add(_T("По радиусу вписанной окружности"));
    this->sizeType->Add(_T("По длине стороны"));
    this->sizeType->SetCurrentByIndex(this->getModel()->sizeType);
}

void MenuBase::initSize(size_t id) {
    BSTR name = _T("Величина размера");
    this->size = this->collection->Add(ControlTypeEnum::ksControlEditLength);
    this->size->Name = name;
    this->size->Hint = name;
    this->size->Tips = name;
    this->size->Id = id;
    this->size->Value = this->getModel()->size;
}

void MenuBase::initHeight(size_t id) {
    BSTR name = _T("Высота");
    this->height = this->collection->Add(ControlTypeEnum::ksControlEditLength);
    this->height->Name = name;
    this->height->Hint = name;
    this->height->Tips = name;
    this->height->Id = id;
    this->height->Value = this->getModel()->height;
}

void MenuBase::initHasHole(size_t id) {
    BSTR name = _T("Есть отверстие");
    this->hasHole = this->collection->Add(ControlTypeEnum::ksControlTwinSwitcher);
    this->hasHole->Name = name;
    this->hasHole->Hint = name;
    this->hasHole->Tips = name;
    this->hasHole->Label2 = _T("Нет отверстия");
    this->hasHole->Id = id;
    this->hasHole->Value = (this->getModel()->hasHole ? 1 : 2);
}

void MenuBase::initHoleRadius(size_t id) {
    BSTR name = _T("Радиус отверстия");
    this->holeRadius = this->collection->Add(ControlTypeEnum::ksControlEditLength);
    this->holeRadius->Name = name;
    this->holeRadius->Hint = name;
    this->holeRadius->Tips = name;
    this->holeRadius->Id = id;
    this->holeRadius->Value = this->getModel()->holeRadius;
}

void MenuBase::initAngleAlpha(size_t id) {
    BSTR name = _T("Угол наклона α");
    this->angleAlpha = this->collection->Add(ControlTypeEnum::ksControlEditAngle);
    this->angleAlpha->Name = name;
    this->angleAlpha->Hint = name;
    this->angleAlpha->Tips = name;
    this->angleAlpha->Id = id;
    this->angleAlpha->Value = this->getModel()->angleAlpha;
}

void MenuBase::initRoundingRadius(size_t id) {
    BSTR name = _T("Радиус скругления");
    this->roundingRadius = this->collection->Add(ControlTypeEnum::ksControlEditLength);
    this->roundingRadius->Name = name;
    this->roundingRadius->Hint = name;
    this->roundingRadius->Tips = name;
    this->roundingRadius->Id = id;
    this->roundingRadius->Value = this->getModel()->roundingRadius;
}

void MenuBase::initButton(size_t id) {
    BSTR name = _T("Применить");
    this->button = this->collection->Add(ControlTypeEnum::ksControlTextButton);
    this->button->Name = name;
    this->button->Hint = name;
    this->button->Tips = name;
    this->button->Id = id;
    this->button->Value = this->getModel()->button;
}

void MenuBase::setFieldName(IPropertyEditPtr field, BSTR name) {
    field->Name = name;
    field->Hint = name;
    field->Tips = name;
}

void MenuBase::setFieldActive(IPropertyEditPtr field) {
    field->ReadOnly = false;
    field->Enable = true;
    field->Visible = true;
}

void MenuBase::setFieldInactive(IPropertyEditPtr field) {
    field->ReadOnly = true;
    field->Enable = false;
    field->Visible = false;
}

void MenuBase::checkHasHole() {
    if (this->hasHole->Value.intVal == 1) {
        this->setFieldActive(field: this->holeRadius);
    }
    else {
        this->setFieldInactive(field: this->holeRadius);
    }
}

```