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

#include <PArray.h>
#include <SafeArrayUtils.h>

extern int LibMessage(LPCTSTR str, int flags = MB_OK);
extern IDocumentFramePtr GetActiveFrame(IKompasDocumentPtr& doc);

void SMPBase::init(IApplicationPtr application, HMODULE hmodule) {
    this->application = application;
    this->hmodule = hmodule;
    this->initDocumentParameters();
}

void SMPBase::initDocumentParameters() {
    this->active_document = this->application->ActiveDocument;
    this->k_document_3d = IKompasDocument3DPtr(this->active_document);
    this->doc3D = ksGetActive3dDocument();

    IPartPtr p = NULL;
    if (this->doc3D) {
        p = this->doc3D->GetPart(Part_Type::pTop_Part);
    }
    this->part = IPartPtr(p);
    this->part7 = this->k_document_3d->TopPart;
    this->model_container = IModelContainerPtr(this->part7);
}

double SMPBase::getOuterRadius() {
    return this->outerRadius;
}

double SMPBase::getInnerRadius() {
    return this->innerRadius;
}

double SMPBase::getSide() {
    return this->side;
}

double SMPBase::getHeight() {
    return (this->height + this->y_delta);
}

unsigned short SMPBase::getNGonSideCount() {
    unsigned short count = 0;
    if (this->isTriangle())
        count = 3;
    else if (this->isSquare())
        count = 4;
    else if (this->isPentagon())
        count = 5;
    else if (this->isHexagon())
        count = 6;
    return count;
}
```

```
58
59  double SMPBase::getRhombusAngle() {
60      double angle = 0;
61      if (this->isRhombus35())
62          angle = 35;
63      else if (this->isRhombus55())
64          angle = 55;
65      else if (this->isRhombus80())
66          angle = 80;
67      return angle;
68  }
69
70  void SMPBase::getPoint(size_t index, double* x, double* y) {
71      size_t size = this->points.size();
72      if (size && (index >= 0) && (index < size)) {
73          *x = get<0>(this->points[index]);
74          *y = get<1>(this->points[index]);
75      }
76  }
77
78  void SMPBase::getPointFirst(double* x, double* y) {
79      this->getPoint(0, x, y);
80  }
81
82  void SMPBase::getPointLast(double* x, double* y) {
83      size_t last = this->points.size() - 1;
84      this->getPoint(last, x, y);
85  }
86
87  void SMPBase::addPoint(double x, double y) {
88      this->points.push_back(make_tuple(x, y));
89  }
90
91  bool SMPBase::isByInnerRadius() {
92      return (this->sizeType == SizeType::byInnerRadius);
93  }
94
95  bool SMPBase::isByOuterRadius() {
96      return (this->sizeType == SizeType::byOuterRadius);
97  }
98
99  bool SMPBase::isBySide() {
100     return (this->sizeType == SizeType::bySide);
101 }
102
103 bool SMPBase::isTriangle() {
104     return (this->surfaceType == SurfaceType::triangle);
105 }
106
107 bool SMPBase::isSquare() {
108     return (this->surfaceType == SurfaceType::square);
109 }
110
111 bool SMPBase::isPentagon() {
112     return (this->surfaceType == SurfaceType::pentagon);
113 }
114
115 bool SMPBase::isHexagon() {
```

```
116     return (this->surfaceType == SurfaceType::hexagon);
117 }
118
119 bool SMPBase::isRhombus35() {
120     return (this->surfaceType == SurfaceType::rhombus35);
121 }
122
123 bool SMPBase::isRhombus55() {
124     return (this->surfaceType == SurfaceType::rhombus55);
125 }
126
127 bool SMPBase::isRhombus80() {
128     return (this->surfaceType == SurfaceType::rhombus80);
129 }
130
131 bool SMPBase::isTrigon() {
132     return (this->surfaceType == SurfaceType::trigon);
133 }
134
135 bool SMPBase::isCircle() {
136     return (this->surfaceType == SurfaceType::circle);
137 }
138
139 bool SMPBase::isNGon() {
140     return (this->isTriangle() ||
141         this->isSquare() ||
142         this->isPentagon() ||
143         this->isHexagon());
144 }
145
146 bool SMPBase::isRhombus() {
147     return (this->isRhombus35() ||
148         this->isRhombus55() ||
149         this->isRhombus80());
150 }
151
152 bool SMPBase::checkHasHole() {
153     return (this->hasHole && (this->holeRadius > 0));
154 }
155
156 IDrawingContainerPtr SMPBase::getDrawingContainer(IFragmentDocumentPtr ↵
      sketch_document) {
157     IKompasDocument2DPtr doc2d(sketch_document);
158     IViewPtr view = doc2d->ViewsAndLayersManager->Views->ActiveView;
159     IDrawingContainerPtr drawing_container(view);
160     return drawing_container;
161 }
162
163 void SMPBase::addEmbodiment() {
164     IEmbodimentsManagerPtr eMng(this->part7);
165     eMng->TopEmbodiment->IsCurrent = true;
166     long count = eMng->EmbodimentCount;
167     long top_index = eMng->CurrentEmbodimentIndex;
168
169     wchar_t temp_str[11];
170     _ltow(count, temp_str, 10);
171     BSTR embodiment_number = SysAllocString(temp_str);
172     BSTR base_marking = _T("СМП");
```

```
173     BSTR additional_number = _T("1");
174     eMng->AddEmbodiment(
175         top_index, false, base_marking,
176         embodiment_number, additional_number
177     );
178     this->initDocumentParameters();
179 }
180
181 void SMPBase::drawBase() {
182     this->base_plane = this->model_container->AddObject
          (ksObj3dTypeEnum::o3d_planeOffset);
183     this->base_plane->Name = _T("Плоскость основания СМП");
184     this->base_plane->BasePlane = this->part7->GetDefaultObject
          (ksObj3dTypeEnum::o3d_planeXOY);
185     this->base_plane->Offset = this->coordinates.Z;
186     this->base_plane->Update();
187
188     this->base_sketch = this->model_container->Sketchs->Add();
189     this->base_sketch->Name = _T("Основание СМП");
190     this->base_sketch->Plane = this->base_plane;
191     IFragmentDocumentPtr sketch_document = this->base_sketch->BeginEdit
          ();
192     IDrawingContainerPtr drawing_container = this->getDrawingContainer
          (sketch_document);
193     if (this->isCircle()) {
194         this->drawCircle(drawing_container);
195     }
196     else if (this->isNGon()) {
197         this->drawNGon(drawing_container);
198     }
199     else if (this->isTrigon()) {
200         this->drawTrigon(drawing_container);
201     }
202     else if (this->isRhombus()) {
203         this->drawRhombus(drawing_container);
204     }
205     this->base_sketch->EndEdit();
206 }
207
208 void SMPBase::drawCircle(IDrawingContainerPtr drawing_container) {
209     ICirclePtr figure = drawing_container->Circles->Add();
210     figure->Style = ksCurveStyleEnum::ksCSNormal;
211     figure->Xc = this->coordinates.X;
212     figure->Yc = this->coordinates.Y;
213     figure->Radius = this->size;
214     figure->Update();
215
216     this->addPoint(
217         this->coordinates.X + this->size,
218         this->coordinates.Y
219     );
220 }
221
222 void SMPBase::drawNGon(IDrawingContainerPtr drawing_container) {
223     unsigned short count = this->getNGonSideCount();
224     double angle_vertex = 360.0 / count;
```

```
225     this->base_sketch->Angle = 180.0;
226
227     IRegularPolygonPtr figure = drawing_container->RegularPolygons->Add ↵
          ();
228     figure->Count = count;
229     figure->Style = ksCurveStyleEnum::ksCSNormal;
230     figure->Xc = this->coordinates.X;
231     figure->Yc = this->coordinates.Y;
232     figure->Radius = this->innerRadius;
233     figure->Describe = true;
234     figure->Update();
235
236     double angle_i;
237     for (size_t i = 0; i < count; i++) {
238         angle_i = angle_vertex * i + angle_vertex / 2;
239         this->addPoint(
240             this->outerRadius * cosd(angle_i),
241             this->outerRadius * sind(angle_i)
242         );
243     }
244 }
245
246 void SMPBase::drawTrigon(IDrawingContainerPtr drawing_container) {
247     double delta_max = this->innerRadius / sind(40);
248     double delta_min = this->innerRadius / sind(80);
249
250     this->addPoint(
251         this->coordinates.X,
252         this->coordinates.Y + delta_max
253     );
254     this->addPoint(
255         this->coordinates.X + delta_min * sind(60),
256         this->coordinates.Y + delta_min * sind(30)
257     );
258     this->addPoint(
259         this->coordinates.X + delta_max * sind(60),
260         this->coordinates.Y - delta_max * sind(30)
261     );
262     this->addPoint(
263         this->coordinates.X,
264         this->coordinates.Y - delta_min
265     );
266     this->addPoint(
267         this->coordinates.X - delta_max * sind(60),
268         this->coordinates.Y - delta_max * sind(30)
269     );
270     this->addPoint(
271         this->coordinates.X - delta_min * sind(60),
272         this->coordinates.Y + delta_min * sind(30)
273     );
274
275     IPolyLine2DPtr figure = drawing_container->PolyLines2D->Add();
276     figure->Style = ksCurveStyleEnum::ksCSNormal;
277     figure->Closed = true;
278     double x, y;
279     for (size_t i = 0; i < 6; i++) {
```

```
280         this->getPoint(i, &x, &y);
281         figure->AddPoint(i, x, y);
282     }
283     figure->Update();
284 }
285
286 void SMPBase::drawRhombus(IDrawingContainerPtr drawing_container) {
287     double angle = this->getRhombusAngle() / 2;
288     double delta_min = this->innerRadius / cosd(angle);
289     double delta_max = this->innerRadius / sind(angle);
290
291     this->addPoint(
292         this->coordinates.X,
293         this->coordinates.Y + delta_max
294     );
295     this->addPoint(
296         this->coordinates.X + delta_min,
297         this->coordinates.Y
298     );
299     this->addPoint(
300         this->coordinates.X,
301         this->coordinates.Y - delta_max
302     );
303     this->addPoint(
304         this->coordinates.X - delta_min,
305         this->coordinates.Y
306     );
307
308     IPolyLine2DPtr figure = drawing_container->PolyLines2D->Add();
309     figure->Style = ksCurveStyleEnum::ksCSNormal;
310     figure->Closed = true;
311     double x, y;
312     for (size_t i = 0; i < 4; i++) {
313         this->getPoint(i, &x, &y);
314         figure->AddPoint(i, x, y);
315     }
316     figure->Update();
317 }
318
319 void SMPBase::addExtrusion() {
320     IExtrusionPtr extrusion = this->model_container->Extrusions->Add
        (ksObj3dTypeEnum::o3d_bossExtrusion);
321     extrusion->Name = _T("Элемент выдавливания СМП");
322     extrusion->Direction = ksDirectionTypeEnum::dtNormal;
323     extrusion->SetSideParameters(
324         true, ksEndTypeEnum::etBlind,
325         this->getHeight(), 0, true, NULL
326     );
327     extrusion->PutDraftValue(true, this->angleAlpha);
328     extrusion->Sketch = this->base_sketch;
329     extrusion->Update();
330 }
331
332 void SMPBase::addRounding() {
333     if (this->isCircle() || !this->roundingRadius) {
334         return;
```

```
335     }
336     double z1 = this->coordinates.Z;
337     double z2 = z1 + this->getHeight();
338     IFilletPtr fillet = this->model_container->AddObject
          (Obj3dType::o3d_fillet);
339     fillet->Name = _T("Скругление боковых рёбер пластины");
340     fillet->Radius1 = this->roundingRadius;
341     fillet->BuildingType = ksFilletBuildingTypeEnum::ksFilletCircleArc;
342     PArray<IModelObjectPtr> fillet_objects;
343
344     IFeature7Ptr feature(this->part7);
345     _variant_t faces = feature->GetModelObjects(Obj3dType::o3d_face);
346     LPDISPATCH* pFaces = NULL;
347     long flCount, flBound, fuBound;
348     setSafeArrayParameters(&faces, &pFaces, &flCount, &flBound,
          &fuBound);
349     for (long idf = 0; idf < flCount; idf++) {
350         IFacePtr face = pFaces[idf];
351         if (face->BaseSurface3DType ==
              ksMathSurface3DTypeEnum::ksPlane) {
352             double s_x1, s_y1, s_z1, s_x2, s_y2, s_z2;
353             face->GetMathSurface()->GetGabarit(&s_x1, &s_y1, &s_z1,
                  &s_x2, &s_y2, &s_z2);
354             if ((round(s_z1) == round(z1)) &&
355                 (round(s_z2) == round(z2))) {
356                 _variant_t edges = face->GetLimitingEdges();
357                 LPDISPATCH* pEdges = NULL;
358                 long elCount, elBound, euBound;
359                 setSafeArrayParameters(&edges, &pEdges, &elCount,
                      &elBound, &euBound);
360                 for (long ide = 0; ide < elCount; ide++) {
361                     IEdgePtr edge = pEdges[ide];
362                     double c_x1, c_y1, c_z1, c_x2, c_y2, c_z2;
363                     edge->GetMathCurve()->GetGabarit(&c_x1, &c_y1,
                        &c_z1, &c_x2, &c_y2, &c_z2);
364                     if (round(c_z1) != round(c_z2)) {
365                         fillet_objects.Add(new IModelObjectPtr(edge));
366                     }
367                 }
368             }
369         }
370     }
371     fillet->BaseObjects = CreateDispSafeArray(fillet_objects);
372     fillet->Update();
373 }
374
375 void SMPBase::addHole() {
376     if (!this->checkHasHole()) {
377         return;
378     }
379     ISketchPtr hole_sketch = this->model_container->Sketchs->Add();
380     hole_sketch->Name = _T("Профиль отверстия СМП");
381     hole_sketch->Plane = this->base_plane;
382     IFragmentDocumentPtr sketch_document = hole_sketch->BeginEdit();
383     IDrawingContainerPtr drawing_container = this->getDrawingContainer
```

```
        (sketch_document);
384     ICirclePtr figure = drawing_container->Circles->Add();
385     figure->Style = ksCurveStyleEnum::ksCSNormal;
386     figure->Xc = this->coordinates.X;
387     figure->Yc = this->coordinates.Y;
388     figure->Radius = this->holeRadius;
389     figure->Update();
390     hole_sketch->EndEdit();
391
392     ICutExtrusionPtr cut_extrusion = this->model_container->Extrusions-
          >Add(ExtrusionType: ksObj3dTypeEnum::o3d_cutExtrusion);
393     cut_extrusion->Name = _T("Отверстие СМП");
394     cut_extrusion->SetSideParameters(
395         Normal: false, ExtrusionType: ksEndTypeEnum::etThroughAll,
396         Depth: this->getHeight(), DraftValue: 0, DraftOutward:
                true, DepthObject: NULL
397     );
398     cut_extrusion->Sketch = hole_sketch;
399     cut_extrusion->Update();
}
```