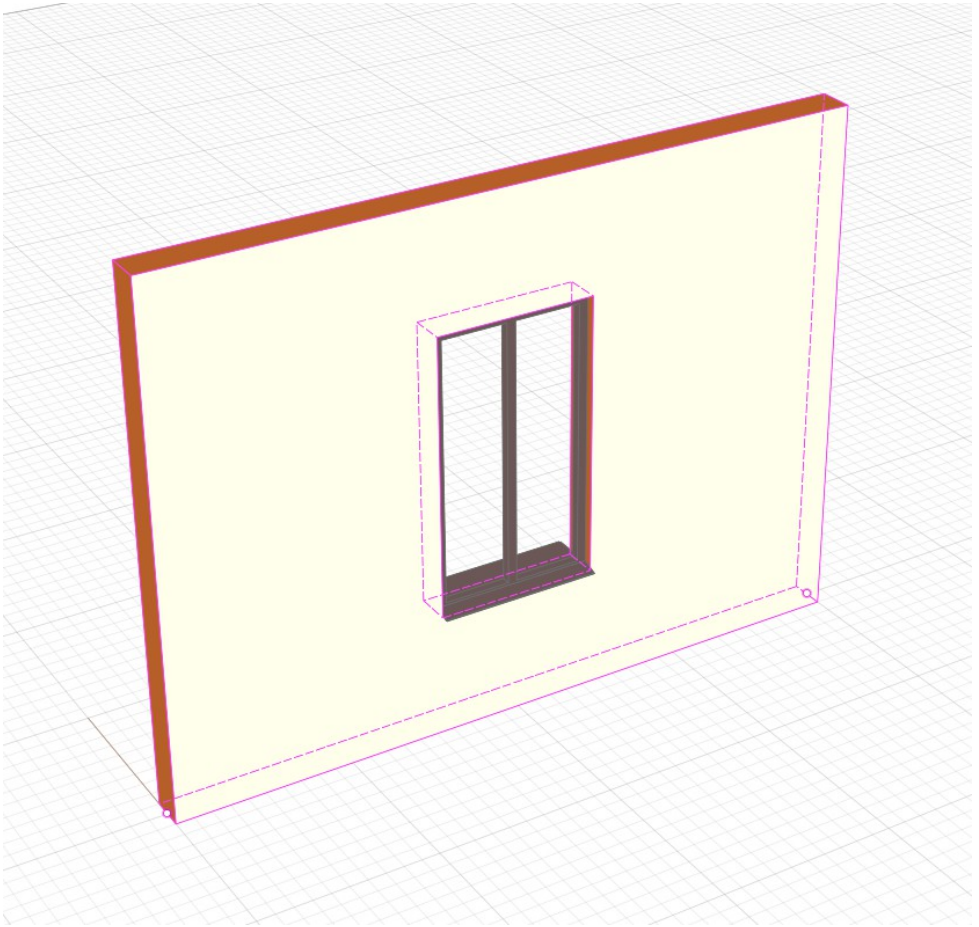
Использование именования и пользовательских атрибутов для разметки частей объектов предметной области

Владислав Монахов
monahov@rengabim.com
Renga Software

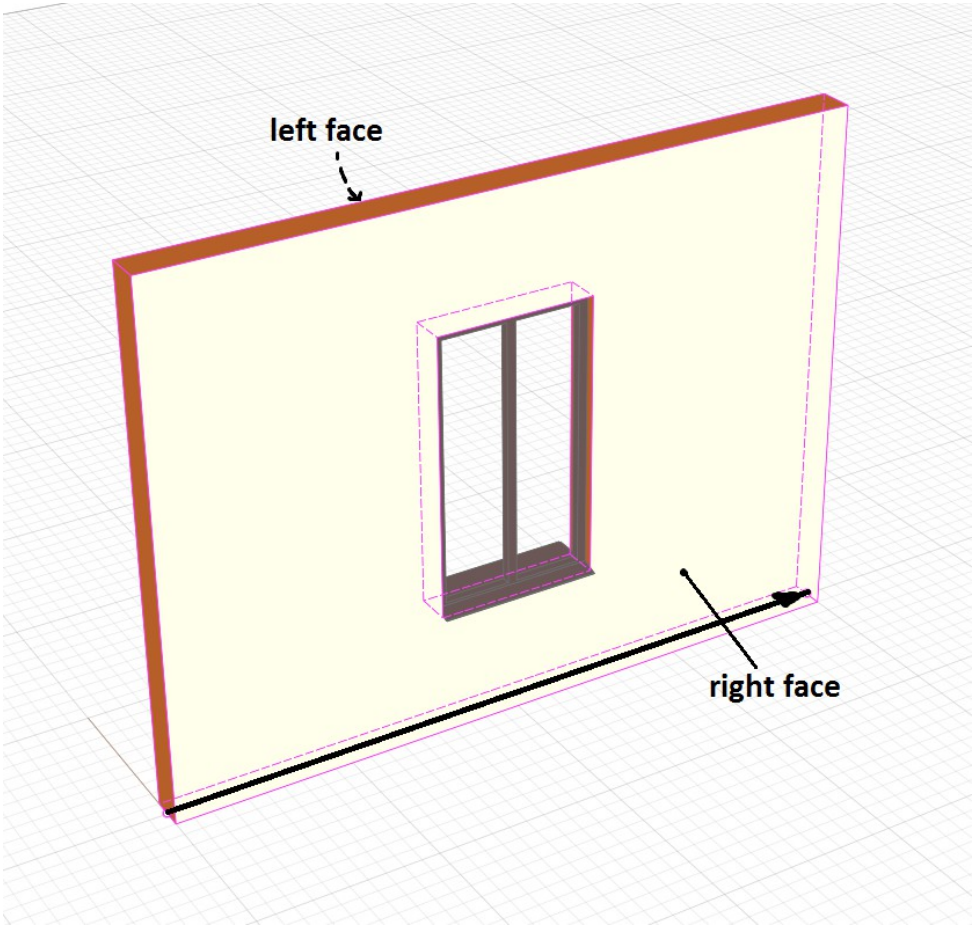
Renga®

Задача: получение площади боковой поверхности стены



Renga®

Площадь боковой поверхности можно определить как сумму площадей правой и левой граней тела стены

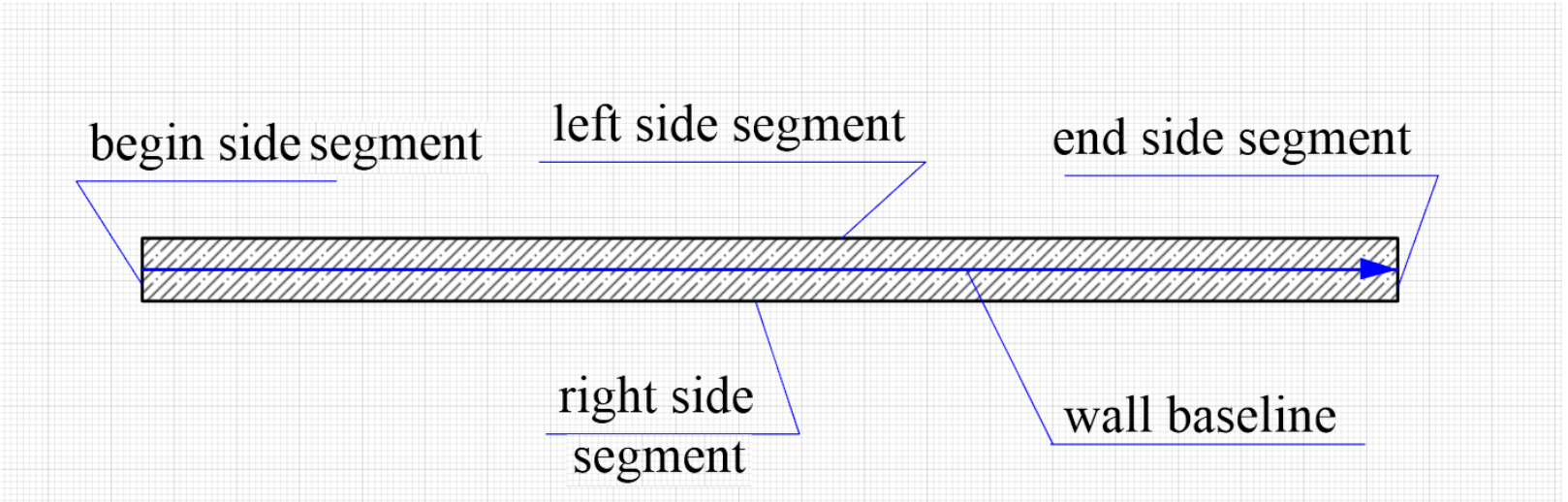


Renga®

Как определить «левую» и «правую» грани стены?

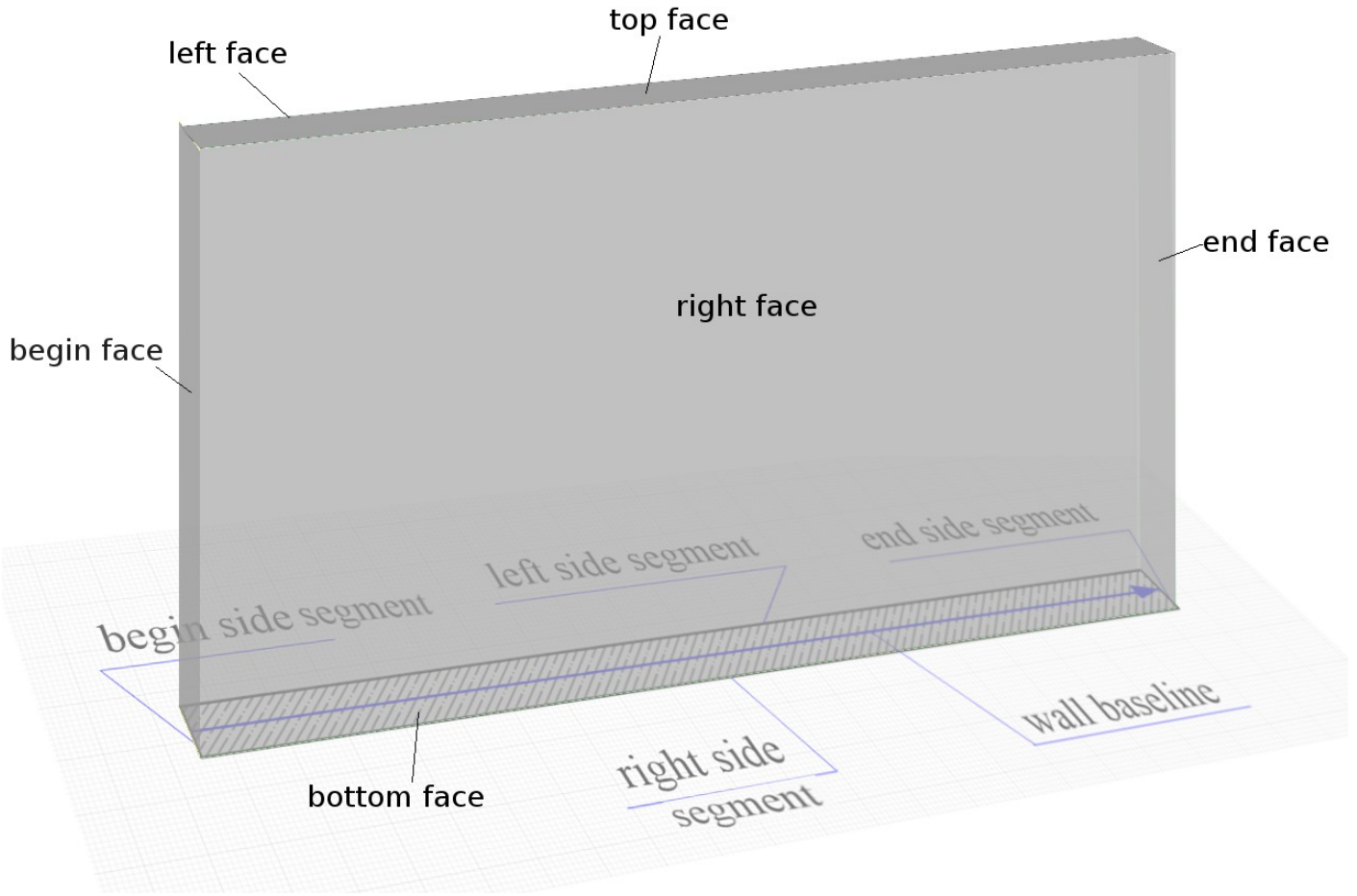
Чтобы ответить на этот вопрос рассмотрим процесс создания тела стены.

В 2D представлении стена — замкнутый контур.
Геометрический смысл каждого сегмента с точки зрения предметной области нам известен:



Renga

Тело получается выдавливанием контура (ExtrusionSolid)



Как сопоставить сегменты контура с полученными гранями тела?

- При создании тела каждая грань получает имя.
- Имена генерирует экземпляр класса **MbSNameMaker**.
- Имя грани генерируется на основе имени сегмента, из которого она создается.

Дадим имена для каждого сегмента:

```
// контур для выдавливания
MbContour* pMyContour = ...; // наш контур, данный свыше

// дадим имя каждому сегменту соответственно их положению в контуре
SimpleNameArray segmentNames;
segmentNames.Add(SimpleName(rightSegmentName)); // 0
segmentNames.Add(SimpleName(endSegmentName)); // 1
segmentNames.Add(SimpleName(leftSegmentName)); // 2
segmentNames.Add(SimpleName(beginSegmentName)); // 3

pMyContour->SetSegmentsNames(segmentNames);
```

Renga®

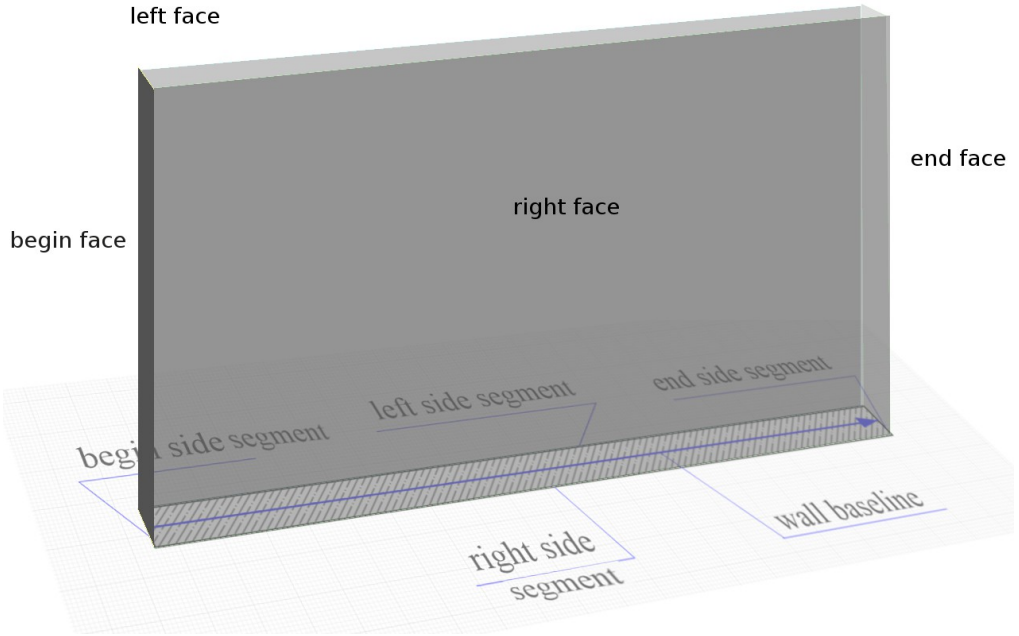
Создаем тело выдавливания:

```
RPArray<MbContour> contours;
contours.Add(pMyContour);
auto spCurveData = std::make_shared<MbSweptData>(..., contours);

// именователь
MbSNameMaker namer(...);

// создание тела выдавливания
MbSolid* pSolid = nullptr;
ExtrusionSolid(*spCurveData, ..., namer, ..., pSolid);
```

Получим имена для всех боковых граней тела:

MbName **rightFaceName, endFaceName, leftFaceName, beginFaceName**;

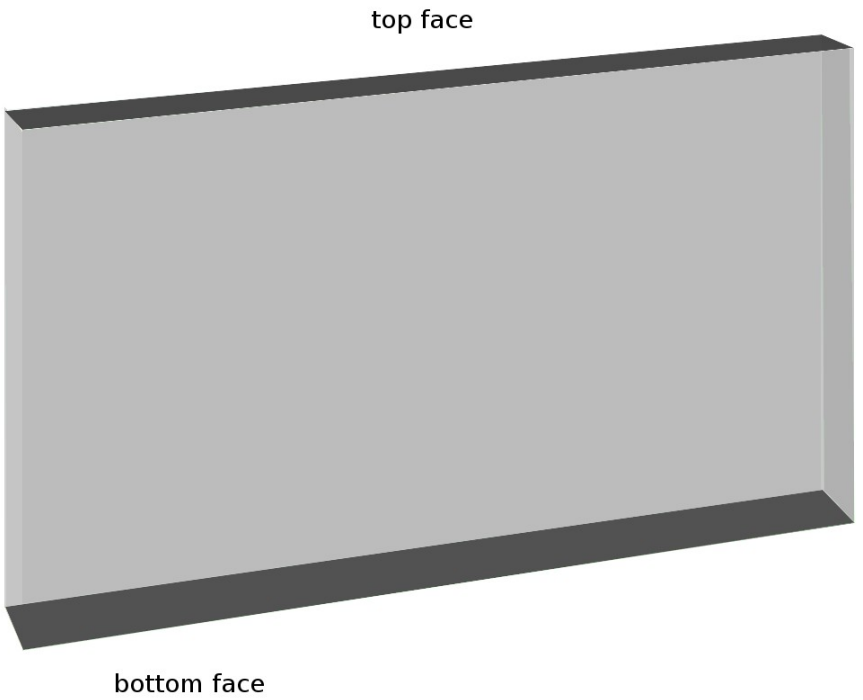
namer.SetFaceName(**rightFaceName**, **rightSegmentName**, ...);
namer.SetFaceName(**endFaceName**, **endSegmentName**, ...);
namer.SetFaceName(**leftFaceName**, **leftSegmentName**, ...);
namer.SetFaceName(**beginFaceName**, **beingSegmentName**, ...);



Renga®

Получим имена для всех торцевых граней тела:

```
MbName topFaceName, bottomButtName;
namer.SetButtFaceName(topFaceName, MbSNameMaker::i_SidePlus);
namer.SetButtFaceName(bottomButtName, MbSNameMaker::i_SideMinus);
```



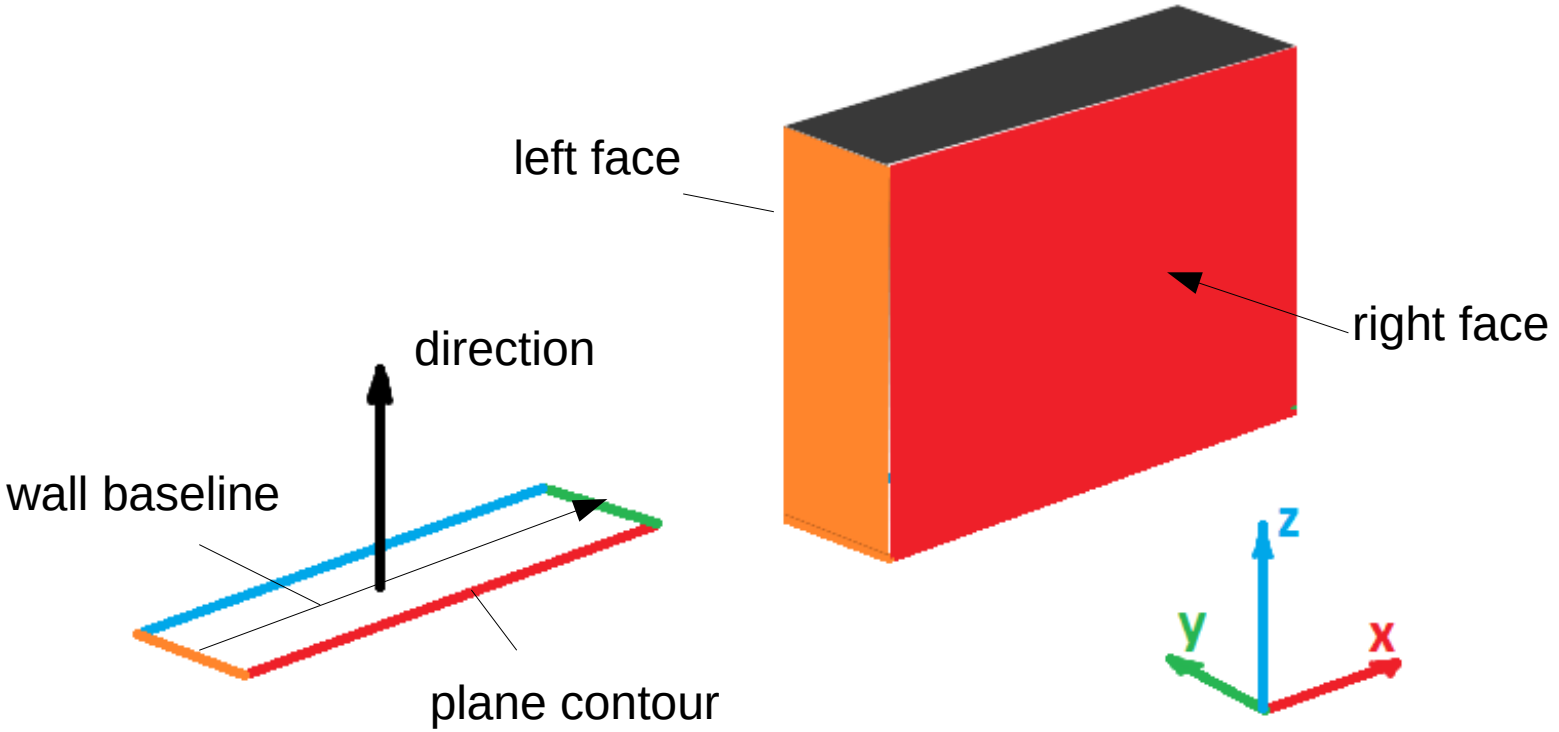
Renga®

По имени мы можем найти нужную нам грань:

```
MbFace* pLeftFace   = pSolid->FindFaceByName(leftFaceName);
MbFace* pRightFace = pSolid->FindFaceByName(rightFaceName);
```

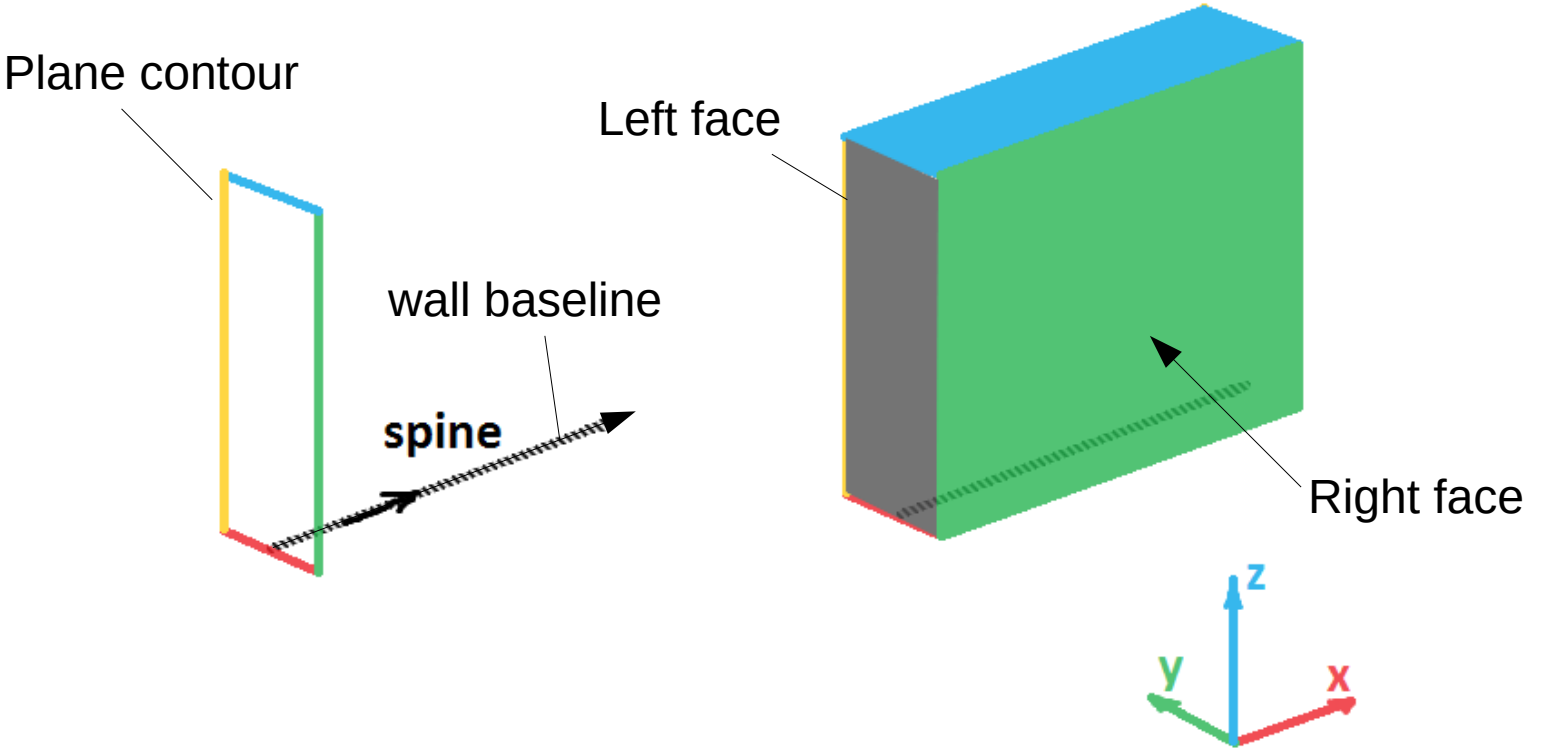
Геометрически одно и тоже тело можно получать множеством способов.
При этом в зависимости от метода создания тела интересующие нас грани могут быть как боковыми так и торцевыми.

Вертикально выдавленный контур (ExtrusionSolid)



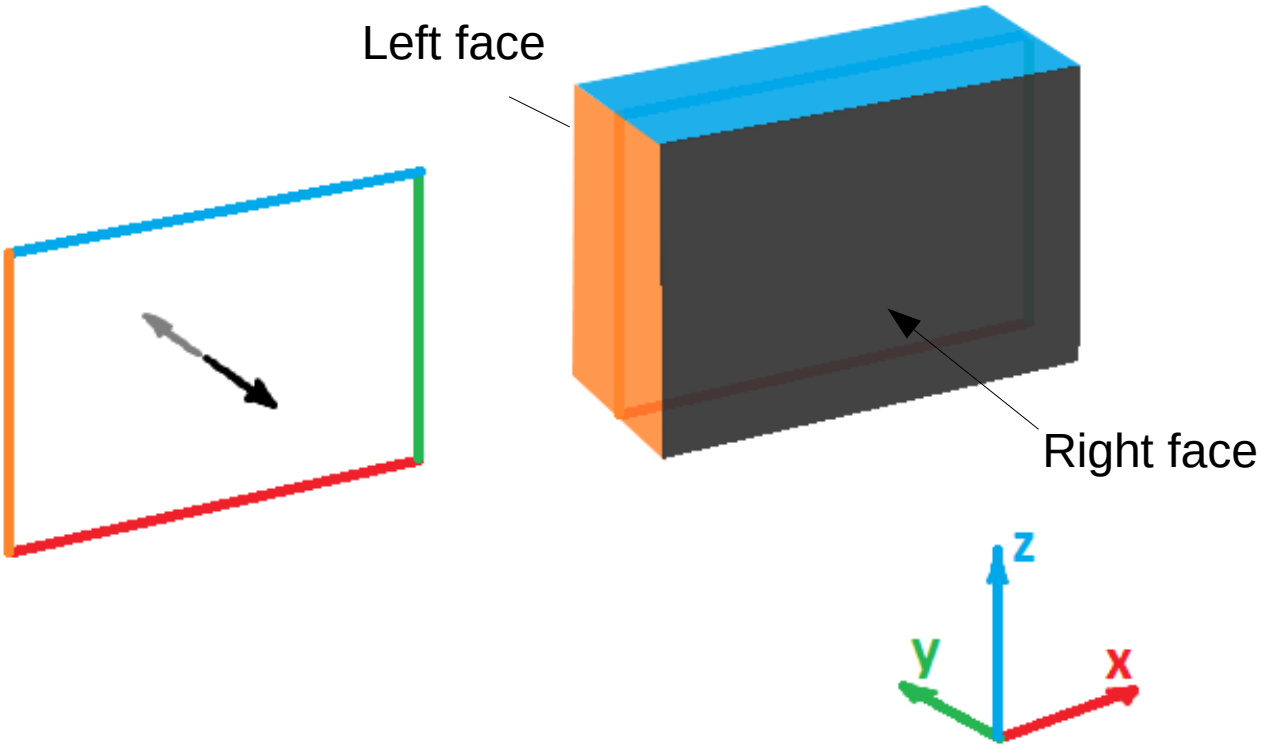
Renga

Контур, выдавленный вдоль пути (EvolutionSolid)



Renga®

«Утолщенный» контур (ThinSolid)



Renga®

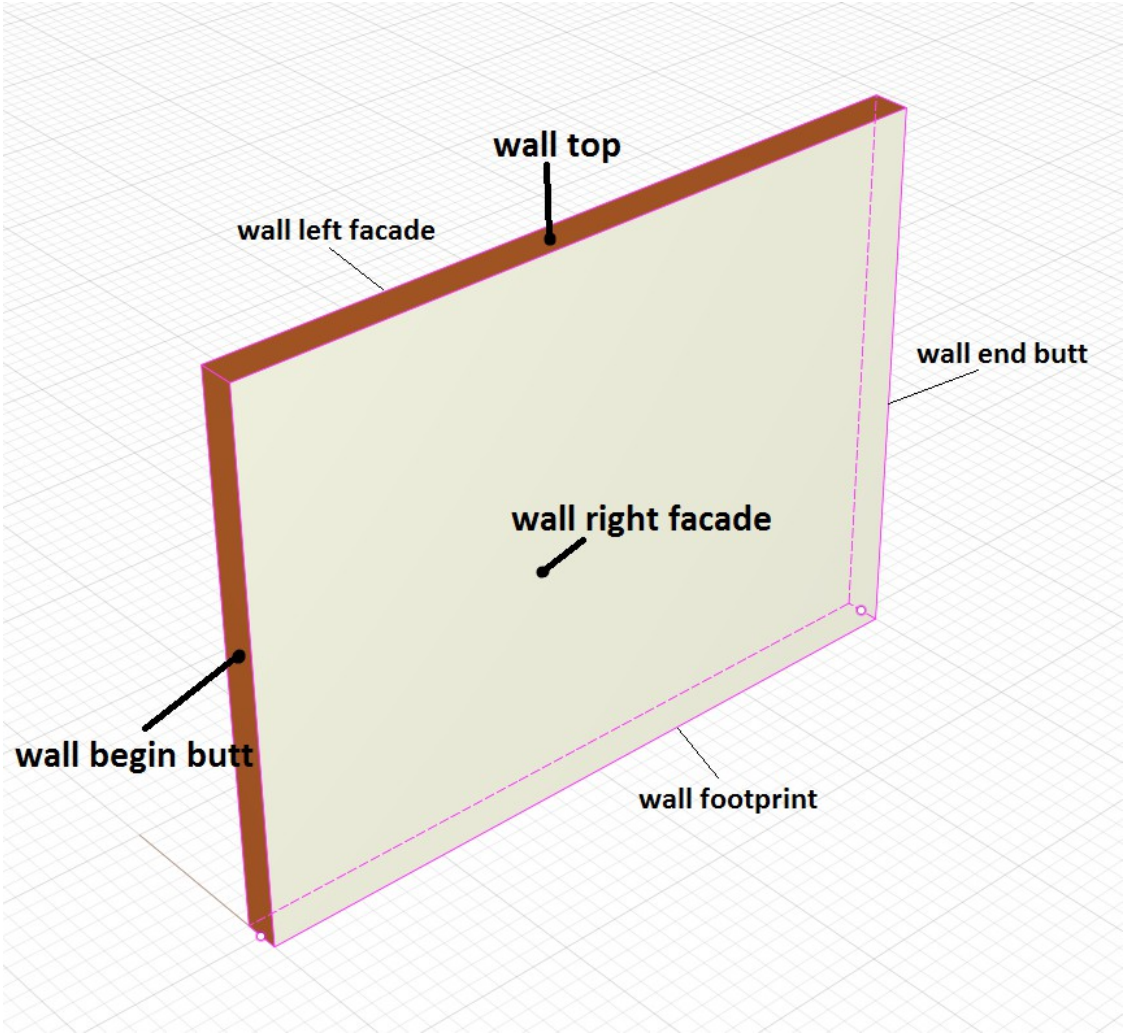
Чтобы абстрагироваться от способа создания тела, мы будем хранить разметку тела в виде пользовательских атрибутов.

```
enum PartTypeEnum // названия частей в предметной области
{
  WallLeftFacade = 0,
  WallRightFacade,
  WallBeginButt,
  WallEndButt,
  WallTop,
  WallFootprint
}

class CPartTypeAttr : public MbUserAttribute
{
public:
  CPartTypeAttr(PartTypeEnum partType)
    : m_partType(partType)
  {...}

  PartTypeEnum getPartType() const {
    return m_partType;
  }

private:
  PartTypeEnum m_partType;
}
```

Вернемся к методу создания тела стены.

```
// узнаем имя грани, полученной из сегмента с именем leftSegmentName
MbName leftFaceName;
namer.SetFaceName(leftFaceName, leftSegmentName, ...);

// найдем грани по имени
MbFace* pLeftFace = pSolid->FindFaceByName(leftFaceName);

// создадим и присвоим атрибут присвоим атрибут
CPartTypeAttr leftFaceAttr(PartTypeEnum::WallLeftFacade);
pLeftFace->AddAttribute(leftFaceAttr);
```

теперь все части стены размечены атрибутами



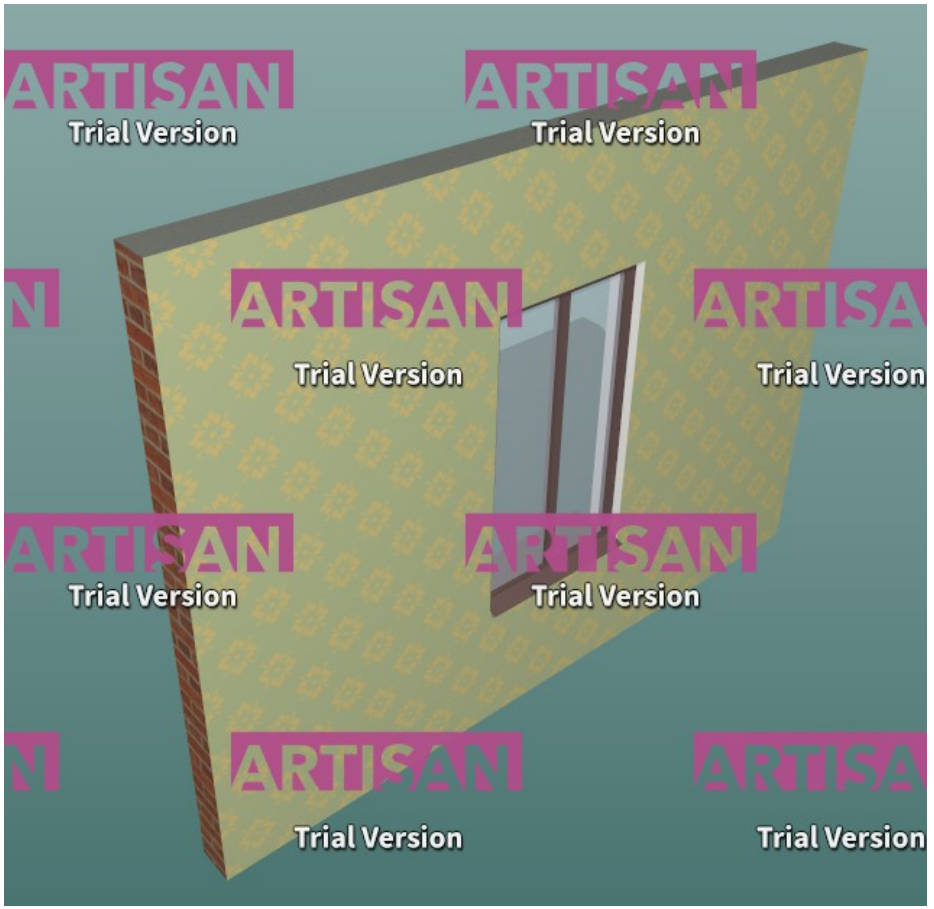
Renga®

```
bool isFacadeFace(MbFace* pFace)
{
  std::vector<MbUserAttribute*> userAttributes;
  pFace->GetUserAttributes(userAttributes, CPartTypeAttr::Uuid());
  _ASSERT(userAttributes.size() == 1);

  auto pPartTypeAttr = static_cast<const CPartTypeAttr*>(userAttributes.front());
  PartTypeEnum partType = pPartTypeAttr->getPartType();
  return (PartTypeEnum::WallLeftFacade == partType)
            || (PartTypeEnum::WallRightFacade == partType);
}

double calculateWallSideArea(const MbSolid& wallSolid)
{
  double sideFaceArea = 0.;
  for (size_t i = 0; i < wallSolid.GetFacesCount(); ++i)
  {
    MbFace* pFace = wallSolid.GetFace(i);
    if (isFacadeFace(pFace))
    {
      sideFaceArea += CalculateArea(*pFace, ...);
    }
  }
  return sideFaceArea;
}
```

Renga®

Для чего можно использовать разметку:

- численных характеристики объектов
- для правильного размещения других объектов (декор, инженерные сети, …)
- визуализация

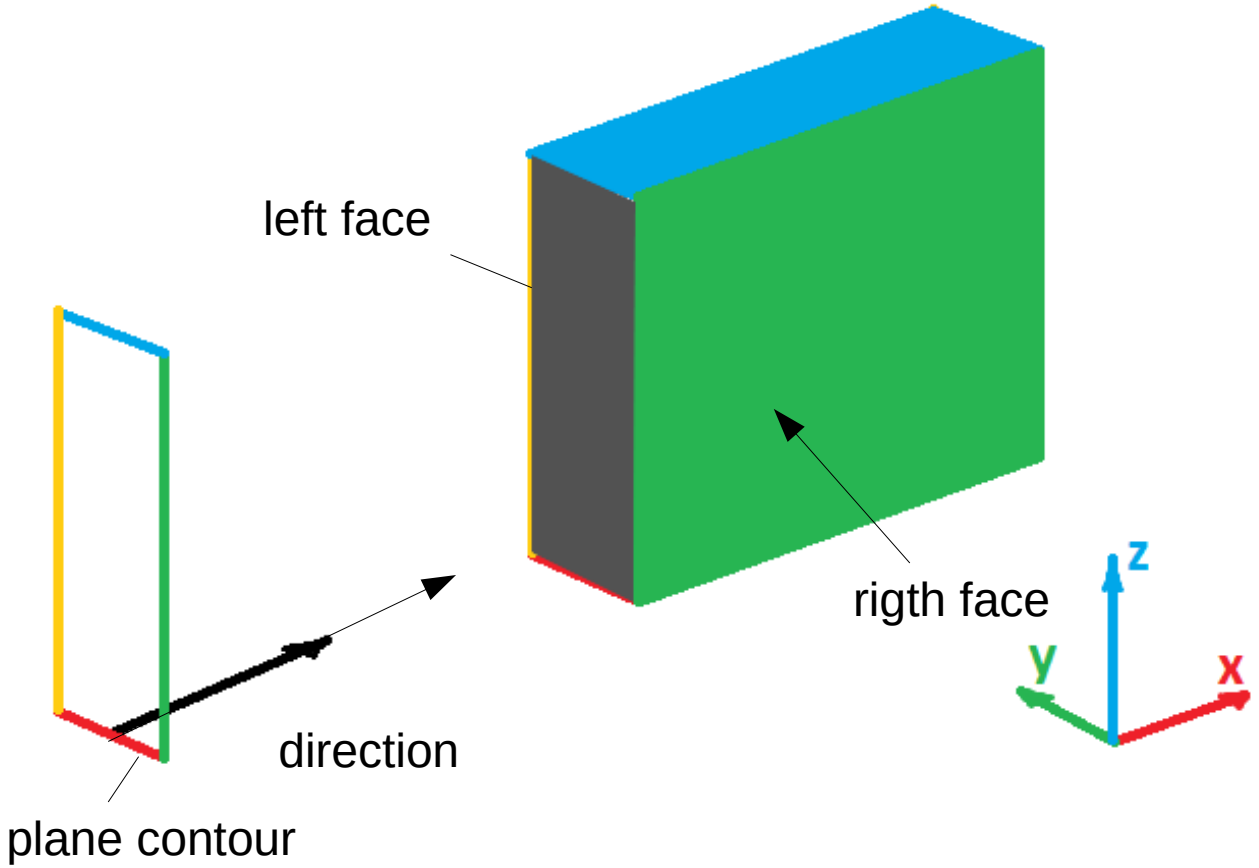


Renga®

Спасибо за внимание

Владислав Монахов
monahov@rengabim.com
Renga Software

Горизонтально выдавленный контур (ExtrusionSolid)



Renga