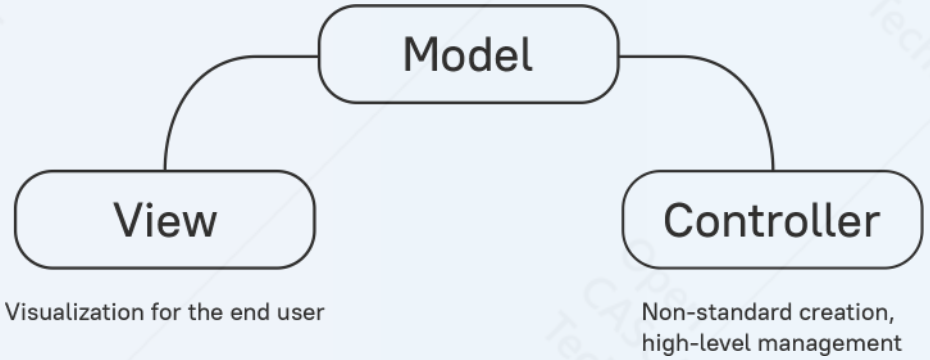
Open CASCADE TECHNOLOGY

Training: Geometry


TRAINING
Open CASCADE TECHNOLOGY
Geometry:
Introduction

MVC. Model-View-Controller

Open CASCADE Technology provides a set of classes the user can apply "as is" or extend by adding custom classes.
The classes dealing with some concept (such as geometrical entities) are usually grouped by the implementation layer:
Model, View, and Controller:



Each layer is implemented in the corresponding package that in turn contain classes and functions.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

MVC. Example

Example: concept of a 2D circle.

Model:

- A data object containing an center and a radius.
- The standard direct constructors: default constructor, constructor with an point and a radius, etc.

Controller:

- To build a circle passing 3 points.
- To build a circle with a center and tangent to a line.

Note: Each controller class has a method to return the abstract data object.

View:

- The object you display in a viewer.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

MVC. Discussion

As usual, there are some advantages and disadvantages.

Advantages:

- Model is perennial: new control classes can be added without changing the model class.
- Model is minimal: instances of controller classes are created when needed during the program execution.

Disadvantages:

- Sometimes, it is difficult to find a class solving a particular problem.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

MVC. Implementation of controller classes

The variety of controller classes implemented in Open CASCADE Technology for geometrical and topological objects include:

- Direct construction (gce_MakeCircle, gce_MakeLin2d).
- Construction by constraints in 2D (GccAna_Circ2d2TanRad).
- Complex construction algorithms: interpolation approximation, projection (GeomAPI package).

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

TRAINING

Open CASCADE TECHNOLOGY

Geometry: Overview

Non-parametric and parametric geometry

Non-parametric geometry

- These types are manipulated by value.
- These classes have no inheritance.

Additional information can be found in documentation:

- Foundation Classes User's Guide.
- Modeling Data User's Guide.
- Modeling Algorithms User's Guide.

Parametric geometry

- Entities from `Geom (Geom2d)` are manipulated by Handle (useful for data sharing), while controller classes are manipulated by value.
- Hierarchy of classes in general follows STEP (ISO 10303) standard.
- Provide methods to go back and forth from `Geom` to `gp`

Additional information can be found in documentation:

- Modeling Data User's Guide.
- Modeling Algorithms User's Guide.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Non-parametric geometry

Model classes (two-dimensional classes are available via adding “2d” suffix, `gp_Pnt2d`):

- `gp_Pnt` – Cartesian point.
- `gp_Vec` – Vector.
- `gp_Dir` – Direction (non-null vector with magnitude equal to 1.0).
- `gp_Trsf` – Euclidean transformation. It is possible to set translation, rotation, and scaling independently.
- `gp_Ax1` – Axis. Axis is point plus direction.
- `gp_Lin`, `gp_Circ`, `gp_Elips`, `gp_Hypr`, `gp_Parab`, `gp_Cylinder`, `gp_Sphere`, `gp_Torus` – primitives representing curves and surfaces.

Controller classes:

- Direct construction – `gce_MakeCircle`, `gce_MakeLin`
- Constrained construction (2d only) – `GccAna_Circ2d2TanRad`

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Limitation of non-parametric geometry

Non-parametric geometry provides useful set classes, but there are some principle limitations with them:

- **Typical geometric questions cannot be answered:**
  - What is the value of curvature at this point?
  - What is the tangent vector to curve at this point?
  - What is the minimum Euclidean distance between the curve and the given point?
  - Do these objects intersect?
- **Some objects are infinite, and there is no way to make them finite:**
  - Line, Hyperbola, Parabola.
  - Plane, Cylinder.
- **It is not possible to represent free-form and non-trivial objects:**
  - How to represent aircraft fuselage? (Bezier and B-spline).
  - How to represent offset surface? (normal is required).
  - How to represent sweeping surfaces? (linear extrusion and revolution).

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Parametric geometry

Model classes (two-dimensional classes are available in the `Geom2d` package):

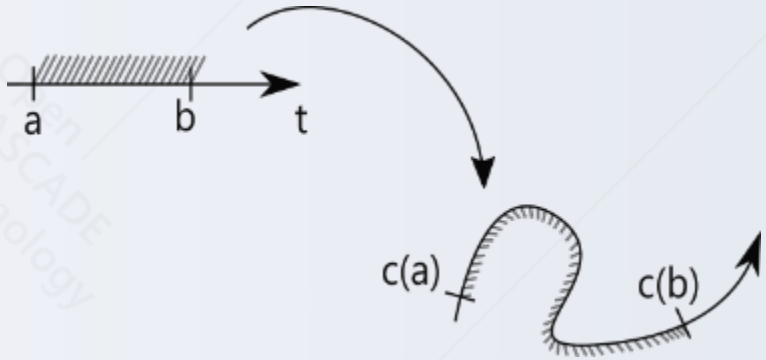
- **Curves – descendants of the** `Geom_Curve`:
  - `Geom_Line`
  - Conics: `Geom_Circle`, `Geom_Ellipse`, `Geom_Hyperbola`, `Geom_Parabola`
  - Free-form: `Geom_BSplineCurve`, `Geom_BezierCurve`
  - `Geom_OffsetCurve`
  - Trimming concept: `Geom_TrimmedCurve`
- **Surface – descendants of the** `Geom_Surface`:
  - Elementary surfaces: `Geom_Plane`, `Geom_CylindricalSurface`, `Geom_SphericalSurface`, `Geom_ToroidalSurface`, `Geom_ConicalSurface`
  - Free-form: `Geom_BSplineSurface`, `Geom_BezierSurface`
  - Sweeping surfaces: `Geom_SurfaceOfLinearExtrusion`, `Geom_SurfaceOfRevolution`
  - `Geom_OffsetSurface`
  - Trimming concept: `Geom_RectangularTrimmedSurface`

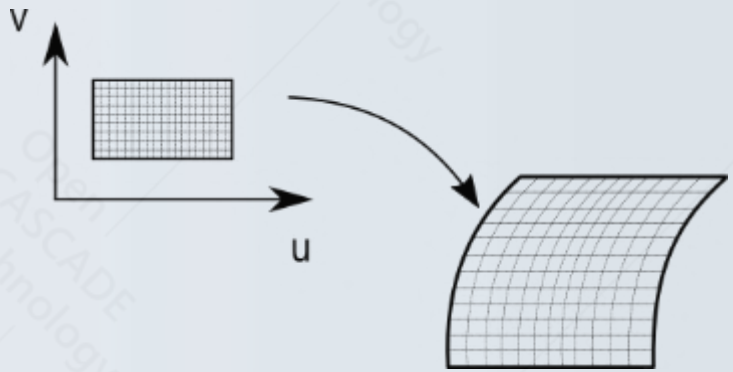
© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Parametric geometry

Controller classes (two-dimensional classes are available via adding “2d” suffix to package name, `gce2d`):

- Direct construction- `gce_MakeCircle, gce2d_MakeCircle`
- Constrained construction (2d only) – `Geom2dGcc_Circl2d3Tan`





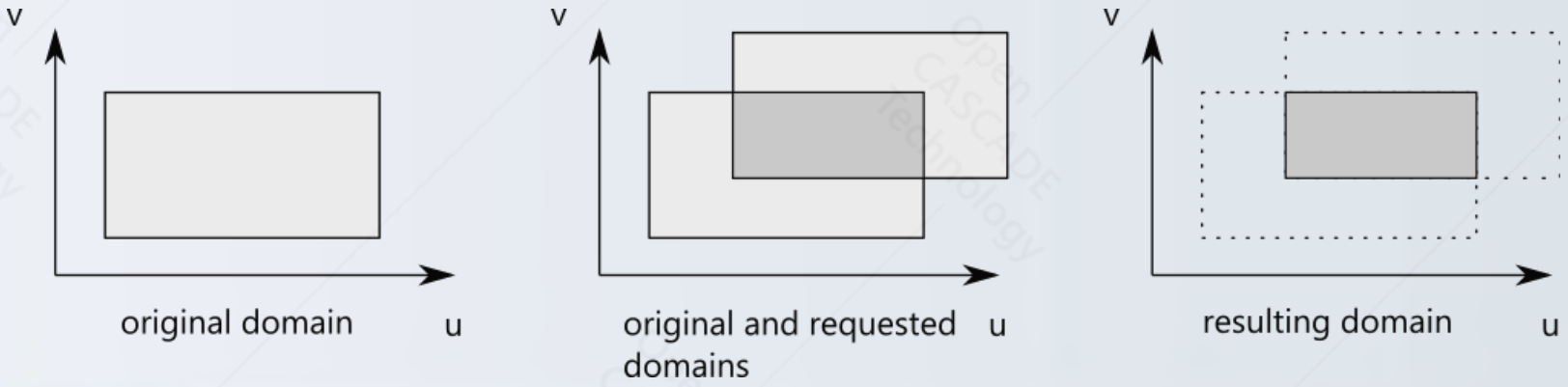
© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Trimming concept

Some parametric objects, like line (`Geom_Line`) or plane (`Geom_Plane`), are infinite as their non-parametric counterparts. How to bound them?

- Curve – bound using starting and finishing parameter – `Geom_TrimmedCurve`
- Surface – bound using rectangular domain – `Geom_RectangularTrimmedSurface`

The OCCT has no protection from surface evaluation outside boundaries. This functionality is used in high-level algorithms such as offset algorithm, but it is recommended to avoid evaluation outside the parametric domain.



© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

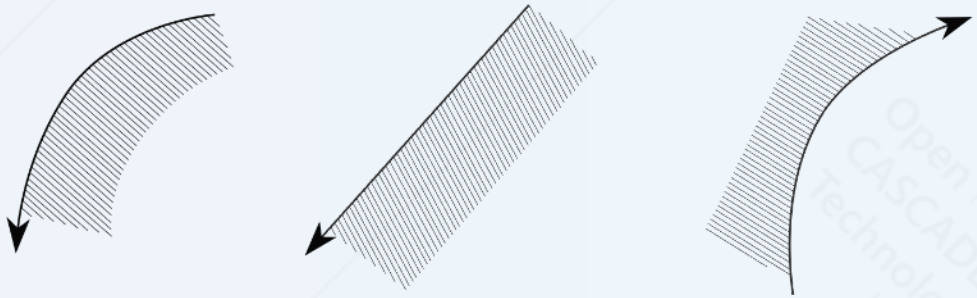
TRAINING

Open CASCADE TECHNOLOGY

Geometry: Constraint geometry

Constraint geometry in 2D

In Open CASCADE Technology, a curve (or a contour) in 2D has an implicit orientation, and the notion of "interior area" is defined.



By convention, the interior of a curve is on the left according to the positive direction of the curve's description. Constraint geometry creation involves qualification of the arguments, in terms of their position relatively to the solution:

Outside - the solution and the argument are outside each other.

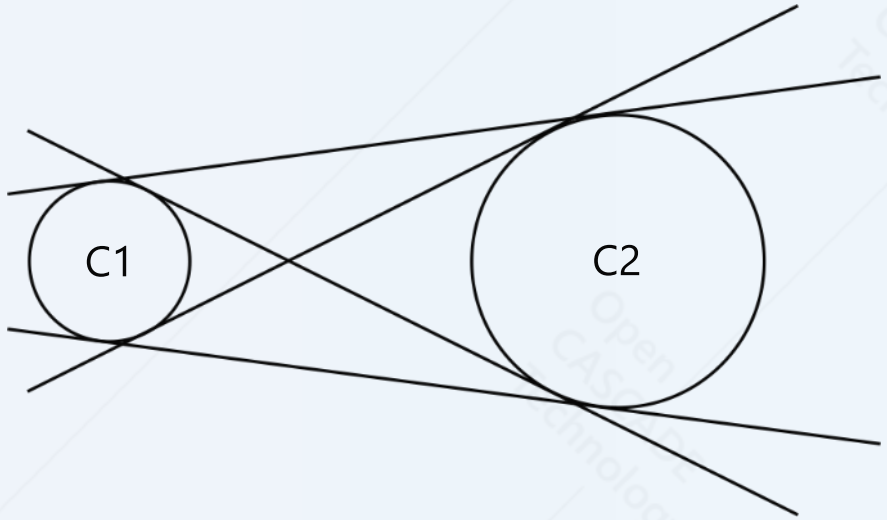
Enclosing - the solution encompasses the argument.

Enclosed - the solution is encompassed by the argument.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

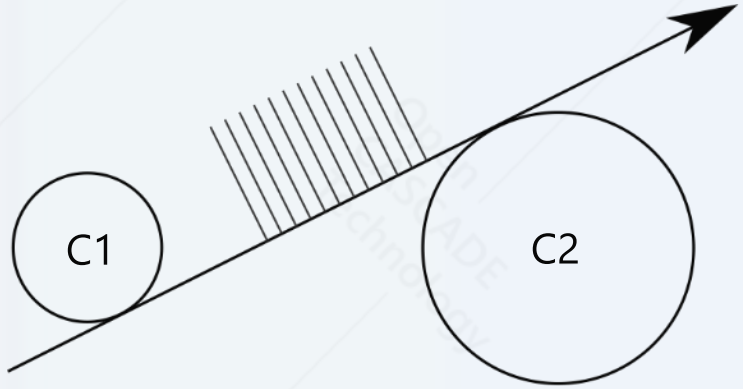
Qualification of arguments

If C1 and C2 are the arguments, the problem of determining a line tangent to C1 and C2 has 4 solutions. These solutions can be reached by `outside` and `enclosing` positioning. `Enclosed` positioning leads to an exception since line cannot be inside a circle.

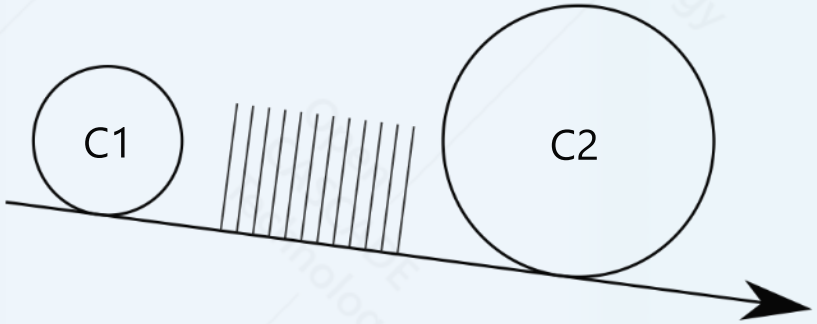


© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Qualification of arguments



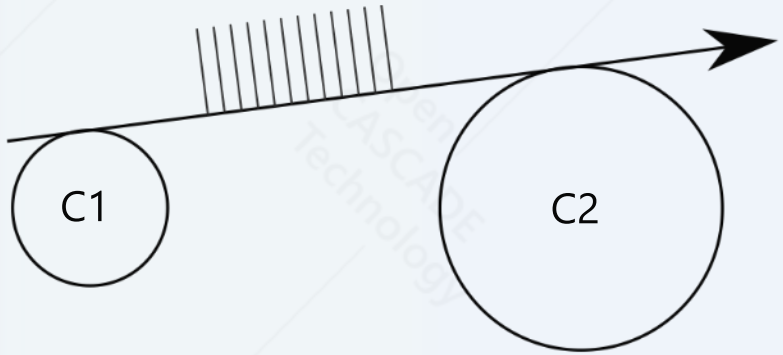
C1 - enclosing
C2 - outside



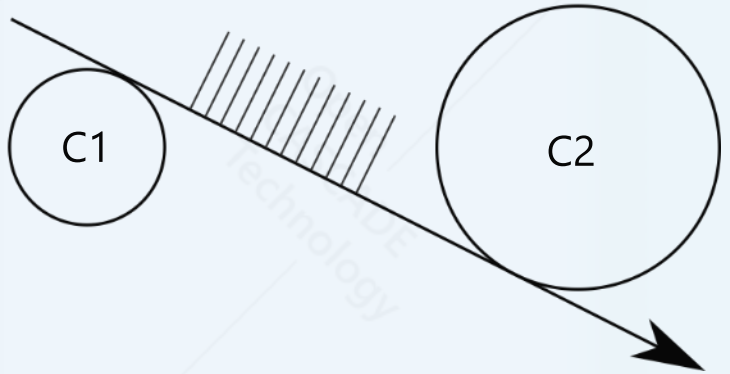
C1 - enclosing
C2 - enclosing

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Qualification of arguments



C1 - outside
C2 - outside



C1 - outside
C2 - enclosing

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Implementation

List of packages used for creation of constraint geometry:

`GccAna` - Algorithm classes for basic geometry.

`Geom2dGcc` - Algorithm classes for advanced geometry.

`GccEnt` - Methods and types for arguments qualification.

Example:

```
#include <GccEnt.hxx>
#include <GccAna_Circ2d2TanRad.hxx>
void ToDo(const gp_Circ2d& aC1, const gp_Circ2d& aC2)
{
  GccAna_Circ2d2TanRad aSolCirc(GccEnt::Outside(aC1), GccEnt::Enclosing(aC2),
                                aRadius, aTolerance);
  if (aSolCirc.IsDone())
  {
    for (Standard_Integer i = 1; i <= aSolCirc.NbSolutions(); i++)
    {
      gp_Circ2d aCircle = aSolCirc.ThisSolution(i);
    }
  }
}
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.


TRAINING
Open CASCADE TECHNOLOGY
Geometry:
API

Floating point: implementation and limitations

Modern computers use IEEE 754 as the way to represent real numbers (alternatives are logarithmic number systems, interval arithmetic, unum / posit). According to the standard real value is represented in the following form:

$$value = significand \cdot base^{exponent}$$

Where significand and exponent are integers. Example:

$$5.4321 = 54321 \cdot 10^{-4}$$

The table below demonstrates bits distribution in common types:

type	Significand(bits)	exponent (bits)
float	24	8
double	53	11

Fixed bits number for precision leads to rounding and presentation errors. So, each algorithm using floating point should be accompanied with tolerance value to cover these errors.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Geometric tolerance and precision

OCCT provides set of geometric tolerances aimed to overcome floating point problems. The are located in `the Precision` package:

type	value	usage
Confusion	1.0e-7	Distances in 3d
PConfusion	Confusion * 0.01	Distances in parametric space
Angular	1.0e-12	Angles equality

Example:

```
// Returns true when the given point is located in origin.
static Standard_Boolean IsOrigin(const gp_Pnt& thePnt)
{
  const gp_Pnt& anOrigin = gp::Origin();
  if (anOrigin.Distance(thePnt) < Precision::Confusion())
    return Standard_True;

  return Standard_False;
}
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

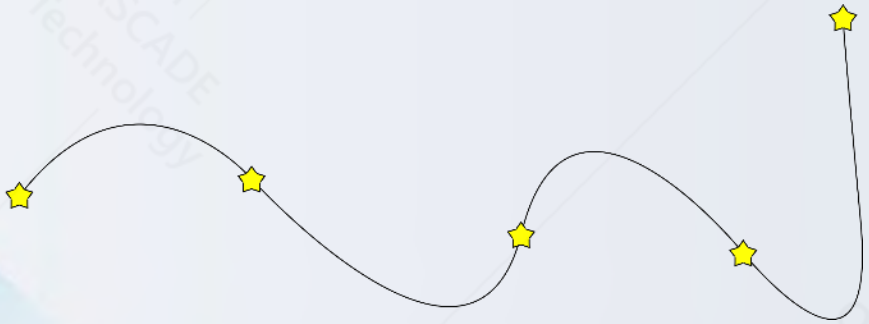
Interpolation vs approximation

Interpolation

Interpolation is finding a polynomial which passes through all points.

The typical problems with them are:

- High order polynomial in result (depends on interpolation algorithm).
- Complex result.
- Oscillations are possible.



Approximation

Approximation is process of polynomial construction which is close to a given set of points but not exactly passes through them. The approximation algorithm has the following drawbacks:

- It is much more computationally intensive comparing to interpolation.
- Requires more efforts to tune parameters comparing to interpolation.

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Curve interpolation

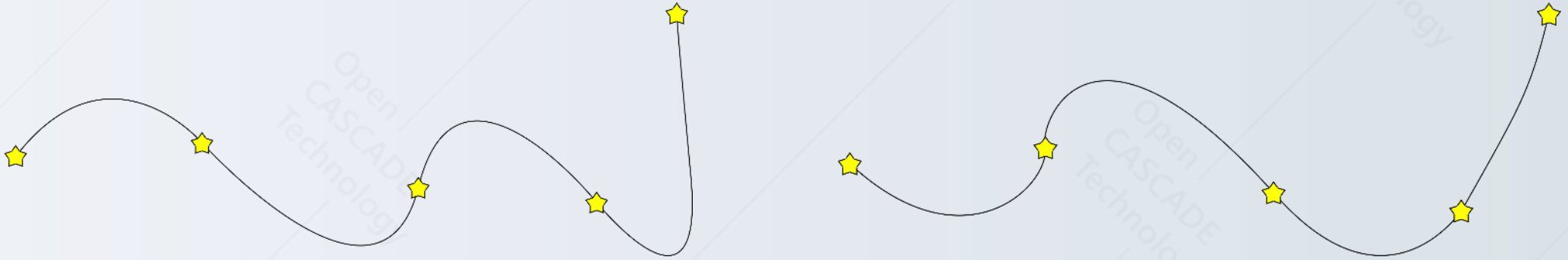
The OCCT provides curve interpolation algorithm. It is possible to run it with and without parameters associated to points. The usage is simple as:

```
// Prepare data to be converted.
Standard_Integer aPos = 1;
Handle(TColgp_HArray1OfPnt) aPnts = new TColgp_HArray1OfPnt(1, 100);

// Fill points
...

// Launch interpolation algorithm.
GeomAPI_Interpolate anInterpolator(aPnts, Standard_False, Precision::Confusion());
anInterpolator.Perform();

// Check done state.
if (!anInterpolator.IsDone())
{
  std::cout << "Error: interpolation failed." << std::endl;
  return 0;
}

Handle(Geom_BSplineCurve) aCurve = anInterpolator.Curve();
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Curve interpolation with tangents

Interpolation problem has infinite solutions:



Particular solution can be chosen using tangents at the starting and final points:

```
GeomAPI_Interpolate anInterpolator(aPnts, Standard_False, Precision::Confusion());
anInterpolator.Load(gp_Vec(1.0, 0.0, 0.0),
                    gp_Vec(0.0, 1.0, 0.0)); // Load tangents.
anInterpolator.Perform();
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Curve approximation

OCCT has approximation built-in curve approximation algorithm. It can be used as follows:

```
// Construct array to be approximated.
TColgp_Array1OfPnt aPnts(1, 100);

// Fill points.
...

GeomAPI_PointsToBSpline anApproximator;
anApproximator.Init(aPnts, 1.0, 0.0, 0.0, 8, GeomAbs_C2, 0.001);

if (!anApproximator.IsDone())
{
  std::cout << "Error: approximation failed." << std::endl;
  return 0;
}

Handle(Geom_Curve) aCurve = anApproximator.Curve();
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Surface interpolation

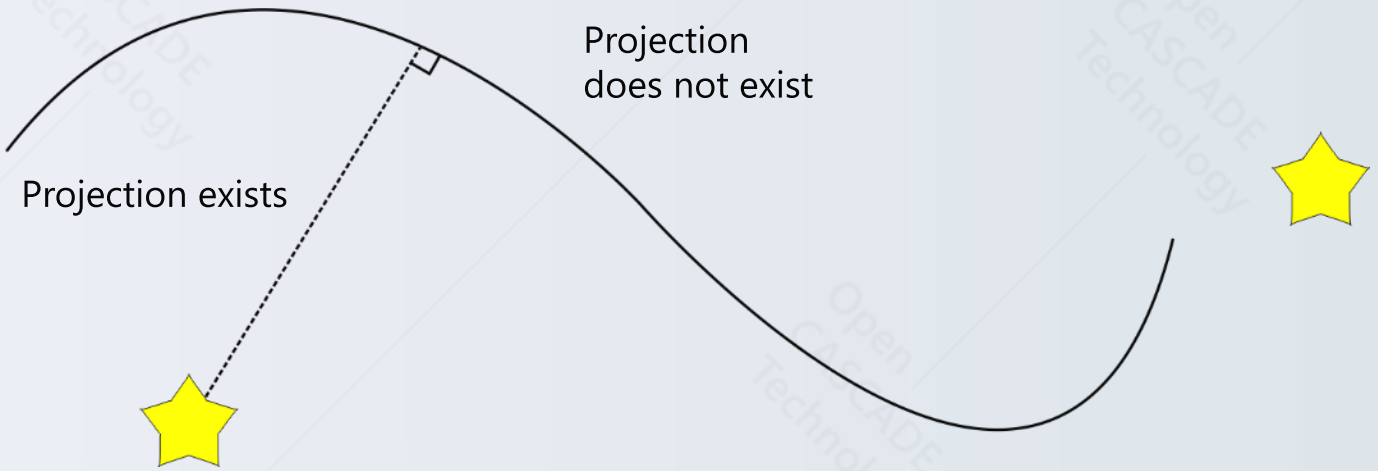
The OCCT supports surface interpolation using input table of points:

```
// Table of points.
TColgp_Array2OfPnt aPnt(1, 3, 1, 3);

// Fill points.
...

// Run interpolator.
GeomAPI_PointsToBSplineSurface aSurfMaker(aPnt);
  if (!aSurfMaker.IsDone())
  {
    std::cout << "Error: Surface construction error" << std::endl;
    return 0;
  }

  Handle(Geom_BSplineSurface) aBsplineSurf = aSurfMaker.Surface();
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Point projection on curve

OCCT contains various projection algorithms such as projection of point on curve or projection curve on surface. Unlike interpolation algorithm, sometimes projection does not exist:



© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

Point projection on surface

Point can be projected not only a curve but also on a surface. The following piece of code demonstrates this functionality:

```
// Prepare surface.
Handle(Geom_Surface) aSurface = ...;

// Project point on surface.
gp_Pnt aPnt = ...;

// Project point on surface.
GeomAPI_ProjectPointOnSurf aProjector;
aProjector.Init(aPnt, aSurface);
gp_Pnt aNPnt = aProjector.NearestPoint();
```

Sometimes, several projections exist. In that case it is necessary to iterate over candidates to get dedicated solution.

```
const Standard_Integer aNbSol = aProjector.NbPoints();
for(Standard_Integer anIdx = 1; anIdx <= aNbSol; ++anIdx)
{
  gp_Pnt aSol = aProjector.Point(anIdx);
}
```

© 2021 Open Cascade. All rights reserved. Confidential - can't be distributed without Open Cascade written consent.

About Open Cascade

It is a software development company which is laser-focused on digital transformation of industries through the use of 3D technologies.

Open Cascade offers a wide range of high-performance proprietary 3D software tools both open-source and commercial. The first ones have been developed, maintained and continuously improved since 2000. Whereas the second ones have been progressively aggregated in the Commercial Platform based on which the company offers creating modern tailor-made industrial solutions that meet even the most sophisticated client's requirements.

Moreover, Open Cascade expands its portfolio by offering end-user industrial software products and delivering software customization and integration services. Open Cascade provides its solutions and services worldwide. The company is a part of the Capgemini's Digital Engineering and Manufacturing Services global business line.

Learn more about Open Cascade at www.opencascade.com

OPEN CASCADE

Backing your path to digital future