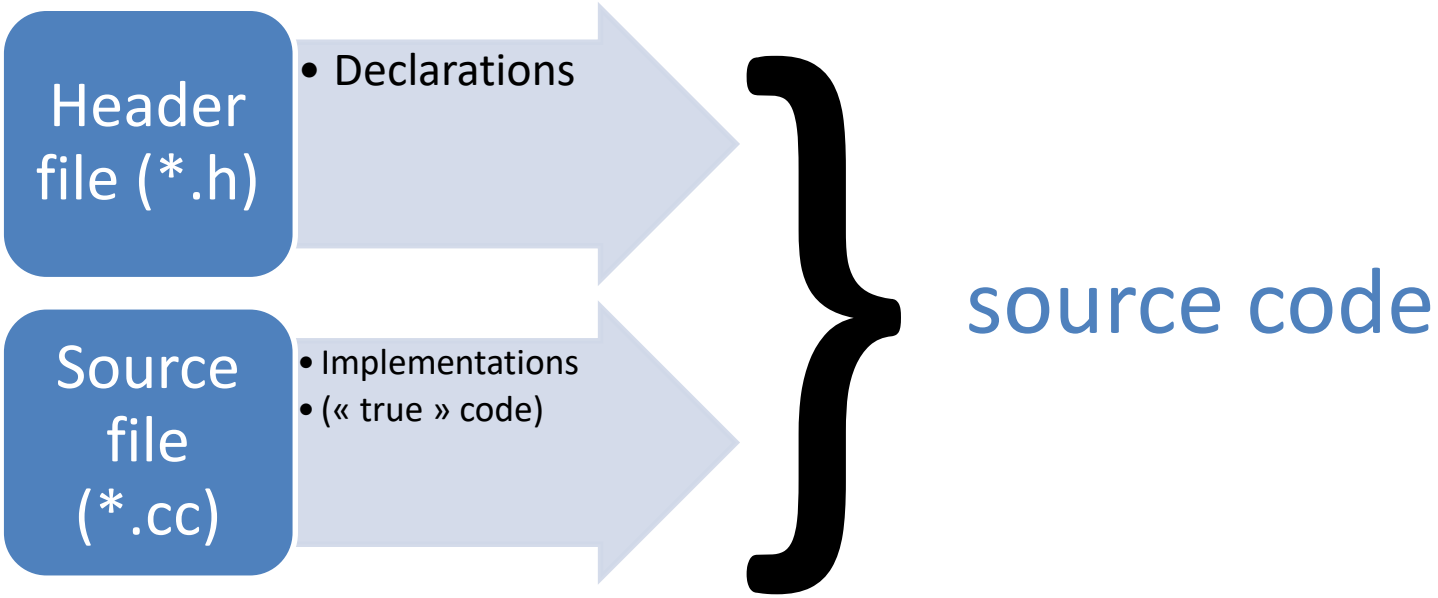
C++ Practical courses

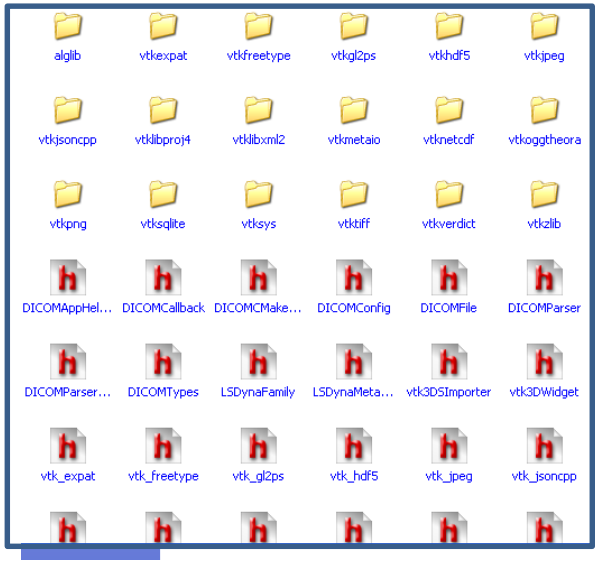
- Who am I?
  - Bolyn Alex(B52/3, +2/…)
    - A.Bolyn@uliege.be
    - Teaching assistant at the A&M department.
- Course web page: http://www.cgeo.ulg.ac.be/CADCG/
  - Schedules, course ressources…
  - Please check-it regularly !

Structure of a C++ project



Structure of a C++ project

Gnurbs lib.



FLTK lib.

CMake

Generation

Project file

Compilation

Executable

Somme references

Book:

"A Tour of C++", Bjarne Stroustrup

Ed. Addison-Wesley (2013)

ISBN 978-0-321-958310

Website: https://en.cppreference.com

(An extensive reference of existing standard functions with examples).

Computer Aided Design

Lagrange polynomials

- The interpolation is represented by the following form :

$$P(u)=\sum_{i=0}^{n-1} P_i\, l_i^p(u) \quad \text{with} \quad l_i^p(u)=\prod_{j=0, i\neq j}^{n-1} \frac{(u-u_j)}{(u_i-u_j)}$$

- Two things worth noting:
  - The curve depends **linearly** on the position of the points
  - It is formed by a weighted sum of **basis functions** that express the influence of each point on the curve

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

Non-reachable from **outside** the class

Reachable from **outside** the class.

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

← Get point on curve of parameter u_

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

← Get point on curve of parameter u_

← Get CP nr. 'which'

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

← Get point on curve of parameter u_

← Get CP nr. 'which'

← Set CP nr. 'which'

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

← Get point on curve of parameter u_

← Get CP nr. 'which'

← Set CP nr. 'which'

← Translate curve by vector 'vec'

Class `nlagrange`

```
class nlagrange : public nparametriccurve
{
protected:
  std::vector<double> pos;
  std::vector<npoint> val;
  int nCP;
public:
  nlagrange(int nCP_);
  virtual int degree(void) const;
  virtual int nb_CP(void) const;
  virtual void add_CP(double u_,const npoint &val_);
  virtual double min_u(void) const;
  virtual double max_u(void) const;
  double LagrangeBasis(int which,double u) const;
  virtual void P(double u_,npoint& ret) const;
  virtual npoint CP(int which) const;
  virtual npoint& CP(int which);
  virtual void set_CP(int which,const npoint& pt);
  void translate(npoint vec);
  virtual double u(int which) const;
  virtual double& u(int which);
};
```

← Parametric values of the control points

← Positions of the control points

← Number of control points

← Constructor

← Get the polynomial degree (read-only)

← Get the number of control points (read-only)

← Add a CP

← Get the minimum u (read-only)

← Get the maximum u (read-only)

← Get point on curve of parameter u_

← Get CP nr. 'which'

← Set CP nr. 'which'

← Translate curve by vector 'vec'

← Get parametric value 'which'

Class `npoint`: usage

- **Constructor**
  - npoint p(0.0, 0.0, 0.0, 1.0); // Euclidean point (0, 0, 0)
- **Get coordinates**
  - double x = p[0]; // Get coordinate x
  - …
  - double w = p[3]; // Get coordinate w (always 1 for the moment)
- **Set coordinates**
  - p[0] = x; // Set coordinate x
  - …
  - p[3] = w; // Set coordinate w

Class `nlagrange`: usage

- **Constructor**
  - nlagrange curve(7); // Create a lagrange curve with 7 CPs.
- **Get/Set parametric values**
  - double u = curve.u(i);
  - curve.u(i) = …;
- **Get/Set control points**
  - npoint cp = curve.CP(i);
  - npoint new_cp(…, …, …, 1); curve.set_CP(i, new_cp);