
Subject: Ideas: Grid lines at "round" dates or values. More options for tooltip.

Posted by [Maginor](#) on Thu, 30 May 2019 10:07:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I have come across a few things that I think could be improved for scientific plotting. I respect that this project is done on people's free time, I am just throwing in some ideas that are of course just my opinion.

I have a graph of daily values. I can use grid unit formatting to get the X axis to display the dates just fine. However, my problem is that the grid lines show up at fairly "random" places. For instance, if I zoom out far enough, I want the grid lines to be at Jan. 1st of every year, but if I zoom in more, I want them to be at the 1st of each month etc (whatever more closely fits the desired amount of grid lines). There could also be some steps in between such as showing a line each half year or each quarter year when appropriate.

This is a graph that the user can freely zoom around in, so it has to be able to update this dynamically.

There does not seem to be any native support for this, and the only way I can see to set the grid lines is by setting their beginning, spacing and count. This will not work for this purpose since Jan. 1st of each year is not equally spaced because of leap years, and similarly months are not equally spaced.

Moreover, I sometimes have data series of just monthly or yearly values, and in those cases I would like the grid lines (of the X axis) to be only where there are data points, never in between.

Another thing I am looking for is that I would like to tell the plot to try to snap grid lines to round values as a default. I.e. I want lines at (0.25, 0.5, 1, 1.25, 1.5, 1.75) rather than (0.05, 0.67, 1.3, 1.92) even if my Y axis happens to range from (0.05 to 1.92). This is something I can do myself just by doing some computations, so it is not a big deal, but it would be nice to have it as an option.

Another thing (Sorry!!):

The tooltip (SHOW_INFO action) in ScatterCtrl shows the x and y position of the cursor. I would like an option for it to show the y value of the graph at that x point (i.e. show (x, f(x)) rather than (x, y)).

If there are multiple graphs, I'm not sure how that should be handled, but maybe either show a list of the values of all the graphs or just show the value of the closest graph to the cursor. If it is a scatter plot rather than a continuous line plot, it would maybe be good to show the value of the closest point on the graph. Similarly, for bar plots it would be good to show the values of the closest bar.

Again, I am very thankful that this library exists at all. I am just throwing in some ideas.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Thu, 30 May 2019 14:44:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Maginor

Thank you, all your comments are valid.

You propose interesting features. Some of them can be solved using `cbModifFormat*` callbacks, but not all. A feature I have never known how to do it is a general and precise zooming algorithm: advices are really acknowledged!

Other feature always waiting to be implemented has been to choose series and series points with a mouse click. If you need this, It will be implemented.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Mon, 12 Aug 2019 14:11:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

I think all my problems with irregular spacing of grid lines would be solved if I could set grid lines explicitly instead of just min, and unit.

So for instance there could be functions

```
ScatterCtrl& SetGridLinesX(Vector<double>& GridLinesX);  
ScatterCtrl& SetGridLinesY(Vector<double>& GridLinesY);
```

And then inside the `WhenZoomScroll` callback, I could just compute the grid lines I want myself and set them using these functions.

I'm not sure what you mean by a zooming algorithm. In what sense?

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Tue, 13 Aug 2019 19:45:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Maginor

The problem with zoom is that the grid spacing would have to be adequate with different zoom levels and scroll. From example:

- If X goes from 0 to 10, spacing could be every 2 (2, 4, 6, 8)
- If X goes from 321 to 501, spacing could be every 40 (361, 401, 441, 481)... or adjusted to more rounded values, every 50, like (350, 400, 450, 500)

Maybe as an intermediate solution, we could add something similar to that you propose, although, instead of returning a `Vector<double>&`, I would propose to add a callback returning `Vector<double>&`, with the values of the grid positions adequate to that level of zoom.

What do you think?

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Wed, 14 Aug 2019 07:38:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

Yes, I think that is a good idea. With custom grid lines you definitely need to update them at every change in zoom level any way, so that would do what I need it to.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Thu, 15 Aug 2019 18:00:17 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Maginor

They are already included and documented, but I have not had time to include them in an example. Please test them:

```
Callback1<Vector<double>&> SetGridLinesX;  
Callback1<Vector<double>&> SetGridLinesY;
```

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Fri, 16 Aug 2019 07:32:51 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thanks a lot! I will test them now.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Fri, 16 Aug 2019 09:35:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi, it almost works now. It is correct initially, but when I pan the plot sideways the grid lines slide in the wrong direction, while the text reticles are correct (and so become unaligned with the grid

lines).

As far as I can see from the code in ScatterDraw::Plot, it seems like the grid lines are drawn with reference to the right of the plot instead of the left?

Here is an example callback where this happens.

```
void PlotCtrl::UpdateDateGridLinesX(Vector<double> &LinesOut)
{
    //InputStartDate is X=0. X resolution is daily

    double XMin = Plot.GetXMin();
    double XRange = Plot.GetXRange();

    Date FirstDate = InputStartDate + (int)XMin;
    int FirstYear = FirstDate.year; //+1?
    Date LastDate = FirstDate + (int)XRange;
    int LastYear = LastDate.year;

    Date IterDate(FirstYear, 1, 1);
    for(int Year = FirstYear; Year <= LastYear; ++Year)
    {
        IterDate.year = Year;
        double XVal = (double)(IterDate - InputStartDate) - XMin;
            if(XVal > 0 && XVal < XRange)
                LinesOut << XVal;
    }
}
```

(Plot is a ScatterCtrl, InputStartDate is just a reference date for X=0).

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Fri, 16 Aug 2019 10:21:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

When I edit ScatterDraw::Plot so that grid x positions are computed in the same way as reticle positions, then things work correctly.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Sat, 17 Aug 2019 20:53:42 GMT

[View Forum Message](#) <> [Reply to Message](#)

Sorry Maginor

X axis zoomed badly. Please check it now.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Mon, 19 Aug 2019 08:21:31 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thanks, it works now!

It's awesome that you were this quick to implement the new feature! I can get exactly the behavior I want out of the graph now.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Tue, 20 Aug 2019 06:00:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Maginor

I am sorry that because of summer time I could support you slower.

A problem to develop this control is to have adequate examples of behaviour of similar controls in other systems, including the source code of control calls if possible. For example, user zoom and scroll behaviour.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Maginor](#) on Tue, 20 Aug 2019 08:31:07 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi, you are probably aware of these, but just in case...

qcustomplot is open source (GPL license, but it should be possible to take general ideas from it without direct copying):

<https://www.qcustomplot.com/>

Then there is the python library matplotlib (backend is implemented in c++)

<https://github.com/matplotlib/matplotlib>

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [koldo](#) on Wed, 21 Aug 2019 06:01:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

Thank you Maginor

I will review qcustomplot. I do not have python.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [deep](#) on Fri, 23 Aug 2019 18:43:19 GMT

[View Forum Message](#) <> [Reply to Message](#)

Audacity plotting is fast. Also it represents correct plot of large data set.
It can serve as good example for large data set.

www.audacityteam.org

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Didier](#) on Sat, 31 Aug 2019 10:49:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Deep and Koldo,

It think I have exactly what you need but it is not directly applicable to ScatterCtrl (I use it with my own plotting ctrl which has different grid/axis management).

- * it does exactly what you want with ordinary numeric grids
- * it does the it's best whith dates (or at least what seemed good for my needs .
since months don't have the same number of days : 29, 30 or 31, it get's quite difficult to have something good looking)
- * it also manages LOG grid steps
- * any custom grid calculation can be plugged in using the `setGridStepCalcBack()` method

NOTE : the nbr of grid steps takes in account the screen size of the plot

I use dedicated classes to manage the grid steps

The strategy is the following:

- * when zoom/scroll/resize ==> recompute the grid which sets an array (`_stepDrawingParams`) containing all the info for the calculated grid steps (position, text to be displayed, and custom data)
- * when drawing needed : the array is used to get the information on the steps

What you are searching for is int the following methods:

```
void stdGridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter& coordConv );  
void log10GridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter& coordConv );  
void dateGridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter& coordConv
```

```
);
void timeGridStepCalcCbk(CLASSNAME& gridStepManager, CoordinateConverter& coordConv
);
```

Here is the class I use for this.

GridStepManager.h

```
#ifndef _GraphCtrl_GridStepManager_h_
#define _GraphCtrl_GridStepManager_h_
```

```
#include <limits>
```

```
namespace GraphDraw_ns
{
```

```
// =====
// GridStepManager CLASS
// =====
```

```
class GridStepData {
public:
    unsigned int tickLevel; // 0:Major tick 1:secondary tick 2: ....
    bool drawTickText;
    Size_<Upp::uint16> textSize;
    TypeGraphCoord stepGraphValue;
    String text;
    Value customParams; // general purpose step parameter holder : filled when calculating
the steps
```

```
    GridStepData()
    : tickLevel(0)
    , drawTickText(true)
    {
    }
};
```

```
class GridStepIterator {
private:
    int _nbSteps; // current step range (according to grid step)
    int _currentStep; // current step number
    CoordinateConverter& _coordConverter;
    GridStepData* const _stepData; // points to steps parameters array (pre-calculated by
GridStepManager)
```

```
public:
```

```
typedef GridStepIterator CLASSNAME;
```

```
GridStepIterator( CoordinateConverter& conv, int nbSteps, GridStepData* stepData, int  
currentStep)
```

```
: _nbSteps(nbSteps)  
, _currentStep(currentStep)  
, _coordConverter(conv)  
, _stepData(stepData)  
{
```

```
GridStepIterator(const CLASSNAME& p)
```

```
: _nbSteps(p._nbSteps)  
, _currentStep(p._currentStep)  
, _coordConverter(p._coordConverter)  
, _stepData(p._stepData)  
{
```

```
public:
```

```
inline bool operator==( const CLASSNAME& v) const { return( _currentStep==v._currentStep ); }  
inline bool operator!=( const CLASSNAME& v) const { return( _currentStep!=v._currentStep ); }  
inline operator TypeScreenCoord() const { return _coordConverter.toScreen( getGraphValue() ); }  
}
```

```
inline operator Size() const { return Size(_stepData[_currentStep].textSize.cx,  
_stepData[_currentStep].textSize.cy); }  
inline operator String() const { return _stepData[_currentStep].text; }  
GridStepData* operator->() { return _stepData + _currentStep; }
```

```
inline TypeGraphCoord getGraphValue() const { return  
_stepData[_currentStep].stepGraphValue; }  
inline const Value& getCustomStepData() const { return  
_stepData[_currentStep].customParams; }  
inline unsigned int getTickLevel() const { return _stepData[_currentStep].tickLevel; }  
inline bool drawTickText() const { return _stepData[_currentStep].drawTickText; }  
inline TypeGraphCoord getGraphRange() const { return  
_coordConverter.getSignedGraphRange(); }  
inline int getStepNum() const { return _currentStep; }  
inline int getNbSteps() const { return _nbSteps; }  
inline bool isFirst() const { return (_currentStep==0); }  
inline bool isLast() const { return (_currentStep==( _nbSteps-1)); }
```

```
inline CLASSNAME& toEnd() { _currentStep = _nbSteps; return *this; }  
inline CLASSNAME& toBegin() { _currentStep = 0; return *this; }
```

```
// ++X
```

```
inline CLASSNAME& operator++()  
{  
++_currentStep;
```

```

return *this;
}

// X++
CLASSNAME operator++(int)
{
    CLASSNAME tmp(*this);
    ++_currentStep;
    return tmp;
}
};

```

typedef Callback2< const GridStepIterator&, String&> TypeFormatTextCbK; // IN: valueIterator, OUT: formatted value

```

class GridStepManager
{
public:
    enum { NB_MAX_STEPS = 100 };
    typedef GridStepManager CLASSNAME;
    typedef GridStepIterator Iterator;
    typedef Callback2<CLASSNAME&, CoordinateConverter&> TypeGridStepCalcCallBack;

```

protected:

```

double    _textSize; // tick Text MAX Size ( can be height or width )
unsigned int _nbMaxSteps; // steps range is : [0, _nbMaxSteps]
unsigned int _nbSteps; // current step range (according to grid step)
unsigned int _currentStep; // current step number
CoordinateConverter& _coordConverter;
bool updateNeeded;

```

GridStepData _stepDrawingParams[NB_MAX_STEPS+1]; // general purpose step parameter holder : filled when calculating the steps

```
TypeGridStepCalcCallBack _updateCbK;
```

```
ChangeStatus coordConvLastStatus;
```

```
TypeGraphCoord GetNormalizedStep(TypeGraphCoord range) const
```

```

{
    int e = 0;
    double s = Upp::normalize(range/(_nbMaxSteps+1), e);
    double sign = 1;
    if (s<0) sign = -1;

    s = sign*s;
    if ( (s>2) && (s<=5) ) { s = 5; }

```

```

else if( (s>1) && (s<=2) ) { s = 2; }
else if( (s>5) && (s<=10) ) { s = 1; e++; }
else { s = 1; }
return sign * s * Upp::ipow10(e);
}

```

```

template <class T>
T GetNormalizedStep(TypeGraphCoord range, const Vector<T>& stepValues) const
{
double s = range/(_nbMaxSteps+1);
T sign = 1;
if (s<0) sign = -1;

s = sign*s;
int c=0;
while (c < stepValues.GetCount()-1) {
if ( (s>stepValues[c]) && (s<=stepValues[c+1]) ) {
s = stepValues[c+1];
return sign*s;
}
++c;
}
if (c==stepValues.GetCount()) s = stepValues[0];
return sign*s;
}

```

```

TypeGraphCoord GetGridStartValue(TypeGraphCoord pstepValue, TypeGraphCoord
pgraphMin) const
{
TypeGraphCoord res;
long double stepValue = pstepValue;
long double graphMin = pgraphMin;
if (graphMin>=0) {
res = ((Upp::int64)(graphMin/stepValue + 1.0 -
std::numeric_limits<TypeGraphCoord>::epsilon()) ) * stepValue;
} else {
res = (((Upp::int64)(graphMin/stepValue) ) * stepValue);
}
if (res < pgraphMin) res = pgraphMin;
return res;
}

```

```

void stdGridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter& coordConv
);
void log10GridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter&
coordConv );
void dateGridStepCalcCbK(CLASSNAME& gridStepManager, CoordinateConverter& coordConv
);

```

```

void timeGridStepCalcCbk(CLASSNAME& gridStepManager, CoordinateConverter& coordConv
);

public:
GridStepManager(CoordinateConverter& coordConv)
: _textSize(30)
, _nbMaxSteps( Upp::min(15, (int)NB_MAX_STEPS) )
, _currentStep( 0 )
, _coordConverter( coordConv )
{
setStdGridSteps();
UpdateGridSteps();
}

GridStepManager(int nbSteps, int currStep, CoordinateConverter& coordConv)
: _textSize(30)
, _nbMaxSteps( Upp::max(5, Upp::min(nbSteps, (int)NB_MAX_STEPS)) )
, _currentStep( currStep )
, _coordConverter( coordConv )
{
setStdGridSteps();
UpdateGridSteps();
}

private:
// explicitly forbidden to use
CLASSNAME& operator=( const CLASSNAME& v) { return *this; }
//GridStepManager()
//: _textSize(30)
//, _nbMaxSteps( 0)//CLASSNAME::NB_MAX_STEPS) )
//, _currentStep( 0 )
//, _coordConverter()
//{}

public:
void setGridStepCalcBack(TypeGridStepCalcCallBack cbk) { _updateCbk = cbk; updateNeeded
= true; }
void setStdGridSteps() { _updateCbk = THISBACK(stdGridStepCalcCbk);
updateNeeded = true; }
void setLogGridSteps() { _updateCbk =
THISBACK(log10GridStepCalcCbk); updateNeeded = true; }
void setTimeGridSteps() { _updateCbk = THISBACK(timeGridStepCalcCbk);
updateNeeded = true; }
void setDateGridSteps() { _updateCbk = THISBACK(dateGridStepCalcCbk);
updateNeeded = true; }

virtual ~GridStepManager() {}

```

```

virtual void UpdateGridSteps()
{
    //RLOGBLOCK("UpdateGridSteps()");
    if ( updateNeeded || coordConvLastStatus.hasChangedUpdate(_coordConverter) ) {
        if ( _textSize > 1.0) _nbMaxSteps = _coordConverter.getScreenRange()/_textSize;
        _updateCbk(*this, _coordConverter);
        updateNeeded = false;
    }
}

void SetNbSteps(int nbSteps)
{
    //_nbMaxSteps = nbSteps;
}

int GetNbSteps() const { return _nbSteps; }

bool SetTextMaxSize(double textSize)
{
    textSize *= 2;
    if ( _textSize < textSize) {
        if ( textSize/_textSize > 1.2) {
            //RLOG("> SetTextMaxSize(>1.2)(" << textSize << ") old _textSize=" << _textSize << "
---UPDATED---");
            _textSize = textSize;
            updateNeeded = true;
            return true;
        }
    } else {
        if (textSize/_textSize < 0.5) {
            //RLOG("< SetTextMaxSize(<0.5)(" << textSize << ") old _textSize=" << _textSize << "
---UPDATED---");
            _textSize = textSize;
            updateNeeded = true;
            return true;
        }
    }
    return false;
}

Iterator End() {
    return GridStepIterator( _coordConverter, _nbSteps+1, _stepDrawingParams, _nbSteps+1);
}

Iterator Begin() {
    return GridStepIterator( _coordConverter, _nbSteps+1, _stepDrawingParams, 0);
}

```

```

};

};
#endif

GridStepManager.cpp
#include "GraphDraw.h"

#define GRIDLOG(a) // LOG(a)

NAMESPACE_UPP
namespace GraphDraw_ns
{
void GridStepManager::stdGridStepCalcCbk(GridStepManager& gridStepManager,
CoordinateConverter& coordConv )
{
TypeGraphCoord gridStepValue = GetNormalizedStep(
(float)coordConv.getSignedGraphRange() ); // DO NOT REMOVE (float) ==> prevents artifacts du
to range precision errors when scrolling (ex: range changes from 14.000000000000052 to
13.99999999999997)
TypeGraphCoord gridStartValue = GetGridStartValue( gridStepValue, coordConv.getGraphMin()
);
_nbSteps = (unsigned int)tabs((coordConv.getGraphMax() - gridStartValue) / gridStepValue);

if (_nbSteps > _nbMaxSteps) {
_nbSteps = _nbMaxSteps;
}

GRIDLOG("stdGrid [" << (int) this << "] =====");
// fill step values ==> used by gridStepIterator
for (unsigned int c=0; c<_nbSteps+1; ++c) {
_stepDrawingParams[c].stepGraphValue = gridStartValue + gridStepValue*c;
_stepDrawingParams[c].drawTickText = 1;// ==> draw tick text
GRIDLOG("stdGrid [" << (int) this << "] - step["<< c <<"] = " <<
_stepDrawingParams[c].stepGraphValue );
}
GRIDLOG("stdGrid [" << (int) this << "] SignedRange = " <<
coordConv.getSignedGraphRange() );
GRIDLOG("stdGrid [" << (int) this << "] getGraphMin = " << coordConv.getGraphMin() );
GRIDLOG("stdGrid [" << (int) this << "] getGraphMax = " << coordConv.getGraphMax() );
GRIDLOG("stdGrid [" << (int) this << "] gridStartValue=" << gridStartValue );
GRIDLOG("stdGrid [" << (int) this << "] gridStepValue = " << gridStepValue );
GRIDLOG("stdGrid [" << (int) this << "] screenRange = " << coordConv.getScreenRange() );
GRIDLOG("stdGrid [" << (int) this << "] screenMin = " << coordConv.getScreenMin() );

```

```

GRIDLOG("stdGrid [" << (int) this << "]  screenMax  =" << coordConv.getScreenMax() );
GRIDLOG("stdGrid [" << (int) this << "]  _nbSteps   =" << _nbSteps );
GRIDLOG("stdGrid [" << (int) this << "]  _nbMaxSteps =" << _nbMaxSteps );
GRIDLOG("stdGrid [" << (int) this << "]  _textSize  =" << _textSize );

}

void GridStepManager::log10GridStepCalcCbk(GridStepManager& gridStepManager,
CoordinateConverter& coordConv )
{
const double rangeFactor = coordConv.getGraphMax()/coordConv.getGraphMin();
if ( rangeFactor < 10 ) {
stdGridStepCalcCbk(gridStepManager, coordConv);
}
else
{
const double logRangeFactor = floor(log10(rangeFactor));
int logStepValue = GetNormalizedStep( (float)logRangeFactor );
if (logStepValue == 0) logStepValue = 1;

TypeGraphCoord gridStartValue = GetGridStartValue(logStepValue,
log10(coordConv.getGraphMin()));
--gridStartValue;
unsigned int nbLogSteps = (log10(coordConv.getGraphMax())-gridStartValue) / logStepValue;
if (nbLogSteps > _nbMaxSteps) {
nbLogSteps = _nbMaxSteps;
}

double logStepDensity = 2;
if ( nbLogSteps > 0 ) logStepDensity = (_nbMaxSteps*logStepValue)/logRangeFactor;

GRIDLOG("=====");
GRIDLOG("log10 getGraphMin  =" << coordConv.getGraphMin() );
GRIDLOG("log10 getGraphMax  =" << coordConv.getGraphMax() );
GRIDLOG("log10 rangeFactor  =" << rangeFactor );
GRIDLOG("log10 logRangeFactor=" << logRangeFactor );
GRIDLOG("log10 logStepValue  =" << double(logStepValue) );
GRIDLOG("log10 gridStartValue=" << gridStartValue );
GRIDLOG("log10 _nbMaxSteps   =" << _nbMaxSteps );
GRIDLOG("log10 nbLogSteps    =" << nbLogSteps );
GRIDLOG("log10 logStepDensity=" << logStepDensity );
GRIDLOG("log10 _textSize     =" << _textSize );

// fill step values ==> used by gridStepIterator
_nbSteps = -1;
for (unsigned int c=0; c<(nbLogSteps+1); ++c)
{

```

```

double stepValue = pow( 10, gridStartValue + logStepValue*c );
double currVal = stepValue;

if ( (currVal > coordConv.getGraphMin()) && (currVal < coordConv.getGraphMax())) {
  ++_nbSteps;
  _stepDrawingParams[_nbSteps].drawTickText = 1; // ==> draw tick text
  _stepDrawingParams[_nbSteps].tickLevel = 0;
  _stepDrawingParams[_nbSteps].stepGraphValue = currVal;
}
currVal += stepValue;

typedef struct {
  bool isStep[9];
  bool displayText[9];
} Log10Steps;

const Log10Steps* logSteps = 0;
if (logStepDensity < 2. || logStepValue>1)
{}
else if (logStepDensity < 3.) {
  static const Log10Steps tmpLogSteps = {
    //1 2 3 4 5 6 7 8 9
    { 1,0,1,0,1,0,1,0,1},
    { 1,0,0,0,1,0,0,0,0}
  };
  logSteps = &tmpLogSteps;
} else if (logStepDensity < 5.) {
  static const Log10Steps tmpLogSteps = {
    //1 2 3 4 5 6 7 8 9
    { 1,1,1,1,1,1,1,1,1},
    { 1,1,0,0,1,0,0,0,0}
  };
  logSteps = &tmpLogSteps;
} else if (logStepDensity < 7.) {
  static const Log10Steps tmpLogSteps = {
    //1 2 3 4 5 6 7 8 9
    { 1,1,1,1,1,1,1,1,1},
    { 1,1,1,0,1,0,0,0,0}
  };
  logSteps = &tmpLogSteps;
} else if (logStepDensity < 11.) {
  static const Log10Steps tmpLogSteps = {
    //1 2 3 4 5 6 7 8 9
    { 1,1,1,1,1,1,1,1,1},
    { 1,1,1,1,1,0,1,0,0}
  };
  logSteps = &tmpLogSteps;
}

```

```

else
{
static const Log10Steps tmpLogSteps = {
//1 2 3 4 5 6 7 8 9
{ 1,1,1,1,1,1,1,1,1},
{ 1,1,1,1,1,1,1,1,1}
};
logSteps = &tmpLogSteps;
}
// fill log steps according to desired pattern configured above
if (logSteps) {
for (unsigned int l=1; l < 9; ++l)
{
if (logSteps->isStep[l]) {
if ( (currVal > coordConv.getGraphMin()) && (currVal < coordConv.getGraphMax())) {
++_nbSteps;
_stepDrawingParams[_nbSteps].drawTickText = logSteps->displayText[l];
_stepDrawingParams[_nbSteps].stepGraphValue = currVal;
_stepDrawingParams[_nbSteps].tickLevel = 1-logSteps->displayText[l];
}
}
currVal += stepValue;
}
}
GRIDLOG("log10 step[" << c << "] = " << _stepDrawingParams[c].stepGraphValue << "
nbLogSteps="<<nbLogSteps);
}
}
}

void GridStepManager::dateGridStepCalcCbk(GridStepManager& gridStepManager,
CoordinateConverter& coordConv )
{
Date dateRange;    dateRange.Set(coordConv.getSignedGraphRange());
Date graphStartDate; graphStartDate.Set(coordConv.getGraphMin());
Date graphEndDate;  graphEndDate.Set(coordConv.getGraphMin());
// =====
//    YEARS
// =====
if (dateRange.year >= 7) {
int yearRange = dateRange.year;
int normalizedYearStep = GetNormalizedStep( yearRange );
if (normalizedYearStep==0) normalizedYearStep = 1;

Date datelater(0,1,1);
datelater.year = GetGridStartValue( normalizedYearStep, graphStartDate.year+1 );
if ( datelater < graphStartDate ) {
GRIDLOG("    #### YEAR STEP_2 :  datelater="<< datelater << "    graphStartDate="<<

```

```

graphStartDate);
    datelter.year += normalizedYearStep;
}

_nbSteps = (unsigned int)tabs((graphEndDate.year - datelter.year) / normalizedYearStep);
if (_nbSteps > _nbMaxSteps) {
    _nbSteps = _nbMaxSteps;
}
else if (_nbSteps==0) _nbSteps = 1;

GRIDLOG("YEAR range=" << yearRange << "years    normalizedStep = " <<
normalizedYearStep << " years    _nbSteps=" << _nbSteps << "    graphStartDate = " <<
datelter);
for (unsigned int c=0; c<= _nbSteps; ++c) {
    Time tmp = ToTime(datelter);
    _stepDrawingParams[c].stepGraphValue = tmp.Get();
    GRIDLOG("  YEAR STEP :  datelter=" << datelter);
    datelter = AddYears(datelter, normalizedYearStep);
}
}
// =====
//    MONTHS
// =====
else if ((dateRange.month >= 7) || (dateRange.year > 0 )) {
    int monthRange = dateRange.year*12 + dateRange.month;
    Vector<double> monthSteps;
    monthSteps << 1 << 2 << 3 << 4 << 6 << 12 << 24 << 48;
    int normalizedMonthStep = GetNormalizedStep( monthRange, monthSteps );
    if (normalizedMonthStep==0) normalizedMonthStep = 1;

    Date datelter(0,1,1);

    int nbMonths = GetGridStartValue( normalizedMonthStep,
graphStartDate.year*12+graphStartDate.month-1 );
    datelter.year = nbMonths/12;
    datelter.month= nbMonths - datelter.year*12 + 1;
    datelter.day = 1;
    GRIDLOG("  MONTH STEP_1 : " << datelter << "    nbMonths=" << nbMonths << "
graphStartDate=" << graphStartDate);
    if ( datelter < graphStartDate ) {
        GRIDLOG("  ##### MONTH STEP_2 :  datelter=" << datelter << "    graphStartDate=" <<
graphStartDate);
        datelter = AddMonths(datelter, normalizedMonthStep);
    }

    _nbSteps = (unsigned int)tabs(((graphEndDate.year-datelter.year)*12 +
(graphEndDate.month-datelter.month)) / normalizedMonthStep);
    if (_nbSteps > _nbMaxSteps) {

```

```

    _nbSteps = _nbMaxSteps;
}
else if (_nbSteps==0) _nbSteps = 1;

    GRIDLOG("MONTH range=" << monthRange << " months    normalizedMonthStep = "<<
normalizedMonthStep << " months    _nbSteps=" << _nbSteps<< "    graphStartDate = "<<
graphStartDate);
    for (unsigned int c=0; c<= _nbSteps; ++c) {
        Time tmp = ToTime(dateltr);
        _stepDrawingParams[c].stepGraphValue = tmp.Get();
        GRIDLOG("    MONTH STEP : "<< dateltr);
        dateltr = AddMonths(dateltr, normalizedMonthStep);
    }
}
// =====
// DAYS
// =====
else {
    TypeGraphCoord gridStartValue;
    Upp::int64 normalizedStep;
    Vector<double> daySteps;
    daySteps << 1 << 2 << 3 << 4 << 5 << 7 << 14 << 28 << 56;
    normalizedStep = GetNormalizedStep( dateRange.Get(), daySteps );
    if (normalizedStep == 0) normalizedStep = 1;
    gridStartValue = GetGridStartValue( normalizedStep, _coordConverter.getGraphMin() );
    _nbSteps = (unsigned int)tabs((graphEndDate.Get() - gridStartValue) / normalizedStep);

    if (_nbSteps > _nbMaxSteps) {
        _nbSteps = _nbMaxSteps;
    }
    else if (_nbSteps==0) _nbSteps = 1;

    GRIDLOG("D/H/M/S ==> _nbSteps=" << _nbSteps << "    gridStepValue=" <<
normalizedStep);
    for (unsigned int c=0; c<_nbSteps+1; ++c)
    {
        _stepDrawingParams[c].stepGraphValue = gridStartValue + normalizedStep * c;
    }
}

void GridStepManager::timeGridStepCalcCbk(GridStepManager& gridStepManager,
CoordinateConverter& coordConv )
{
    enum {
        DAY_sec    = 24*60*60,
        HOUR_sec   = 60*60,
        MINUTES_sec = 60,

```

```

};
Time timeOrigin;  timeOrigin.Set(0);
Time graphStartTime; graphStartTime.Set(coordConv.getGraphMin());
Time graphEndTime;  graphEndTime.Set(coordConv.getGraphMax());
Time timeRange;    timeRange.Set(coordConv.getSignedGraphRange());

GRIDLOG("\n--timeGridStepCalcCbK-- range=" << timeRange << "    GraphMin = "<<
graphStartTime << "    GraphMax = " << graphEndTime);

// =====
//    YEARS
// =====
if (timeRange.year >= 7) {
    Date graphStartDate( graphStartTime.year, graphStartTime.month, graphStartTime.day);
    Date graphEndDate( graphEndTime.year, graphEndTime.month, graphEndTime.day);
    int yearRange = timeRange.year;
    int normalizedYearStep = GetNormalizedStep( yearRange );
    if (normalizedYearStep==0) normalizedYearStep = 1;

    Date datelater(0,1,1);
    datelater.year = GetGridStartValue( normalizedYearStep, graphStartDate.year+1 );
    if ( datelater < graphStartDate ) {
        GRIDLOG("    #### YEAR STEP_2 :  datelater="<< datelater << "    graphStartDate="<<
graphStartDate);
        datelater.year += normalizedYearStep;
    }

    _nbSteps = (unsigned int)tabs((graphEndDate.year - datelater.year) / normalizedYearStep);
    if (_nbSteps > _nbMaxSteps) {
        _nbSteps = _nbMaxSteps;
    }
    else if (_nbSteps==0) _nbSteps = 1;

    GRIDLOG("YEAR range=" << yearRange << "years    normalizedStep = "<<
normalizedYearStep << " years    _nbSteps=" << _nbSteps<< "    graphStartDate = "<<
datelater);
    for (unsigned int c=0; c<= _nbSteps; ++c) {
        Time tmp = ToTime(datelater);
        _stepDrawingParams[c].stepGraphValue = tmp.Get();
        GRIDLOG("    YEAR STEP :  datelater="<< datelater);
        datelater = ToTime(AddYears(datelater, normalizedYearStep));
    }
}
// =====
//    MONTHS
// =====
else if ((timeRange.month >= 7) || (timeRange.year > 0 )) {
    Date graphStartDate( graphStartTime.year, graphStartTime.month, graphStartTime.day);

```

```

Date graphEndDate( graphEndTime.year, graphEndTime.month, graphEndTime.day);
int monthRange = timeRange.year*12+timeRange.month-1 ;
Vector<double> monthStep;
monthStep << 1 << 2 << 3 << 4 << 6 << 12 << 24 << 48;
int normalizedMonthStep = GetNormalizedStep( monthRange, monthStep );
if (normalizedMonthStep==0) normalizedMonthStep = 1;

Date datelater(0,1,1);

int nbMonths = GetGridStartValue( normalizedMonthStep,
graphStartDate.year*12+graphStartDate.month-1 );
datelater.year = nbMonths/12;
datelater.month= nbMonths - datelater.year*12 + 1;
datelater.day = 1;
GRIDLOG("  MONTH STEP_1 : "<< datelater << "    nbMonths=" << nbMonths << "
graphStartDate=" << graphStartDate);
if ( datelater < graphStartDate ) {
  GRIDLOG("  ##### MONTH STEP_2 :  datelater="<< datelater << "    graphStartDate="<<
graphStartDate);
  datelater = AddMonths(datelater, normalizedMonthStep);
}

_nbSteps = (unsigned int)tabs(((graphEndDate.year-datelater.year)*12 +
(graphEndDate.month-datelater.month)) / normalizedMonthStep);
if (_nbSteps > _nbMaxSteps) {
  _nbSteps = _nbMaxSteps;
}
else if (_nbSteps==0) _nbSteps = 1;

GRIDLOG("MONTH range=" << monthRange << " months  normalizedMonthStep = "<<
normalizedMonthStep << " months  _nbSteps=" << _nbSteps<< "    graphStartDate = "<<
graphStartDate);
for (unsigned int c=0; c<= _nbSteps; ++c) {
  Time tmp = ToTime(datelater);
  _stepDrawingParams[c].stepGraphValue = tmp.Get();
  GRIDLOG("  MONTH STEP : "<< datelater);
  datelater = AddMonths(datelater, normalizedMonthStep);
}
}
// =====
// DAY / HOUR / MINUTES / SECONDS
// =====
else {
  TypeGraphCoord gridStartValue;
  Upp::int64 normalizedStep;
  Vector<double> secondsSteps;
  secondsSteps << 1 << 2 << 5 << 10 << 15 << 20 << 30;
  secondsSteps << MINUTES_sec << MINUTES_sec*2 << MINUTES_sec*3 << MINUTES_sec*5

```


[View Forum Message](#) <> [Reply to Message](#)

Oh, wow. That is cool.

I just wrote my own functionality to do this, but it is not as nicely refactored into a reusable library.

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.

Posted by [Didier](#) on Tue, 03 Sep 2019 18:02:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hope it helps

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.

Posted by [Pradip](#) on Tue, 28 Apr 2020 07:03:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Didier and All,

I need to make Gantt chart displaying bars for activities on a timeline, so I'll need the exact functionality Maginor was looking for. Where can I find the GraphCtrl/GraphDraw package?

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.

Posted by [Didier](#) on Fri, 01 May 2020 09:33:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Pradip,

You mean something like this (with time on X axis) ?

I recently added the capacity to display fully custom data (with mouse interactions) just for the same reasons

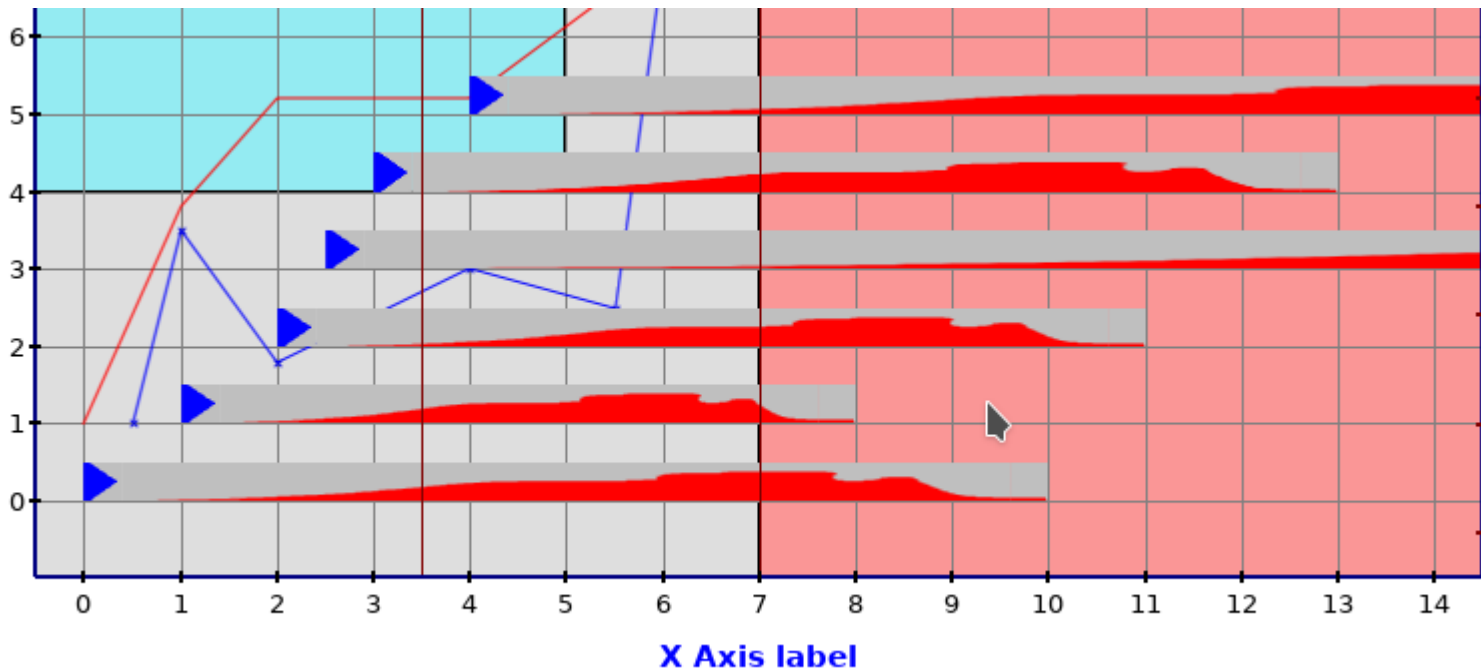
and I am currently working on adding full Chameleon compatibility changes still need to be done :
Layout designer properties do not work anymore (du to Chameleon modifications) made corrections to the property editors

So the Package is still work in progress

I will upload the current state so that you can play around with it

File Attachments

1) [Example.png](#), downloaded 802 times



Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
Posted by [Pradip](#) on Fri, 01 May 2020 10:02:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Didier,

Thanks for replying! Yes exactly, I need date/time on x-axis. I'm trying to make something like this:

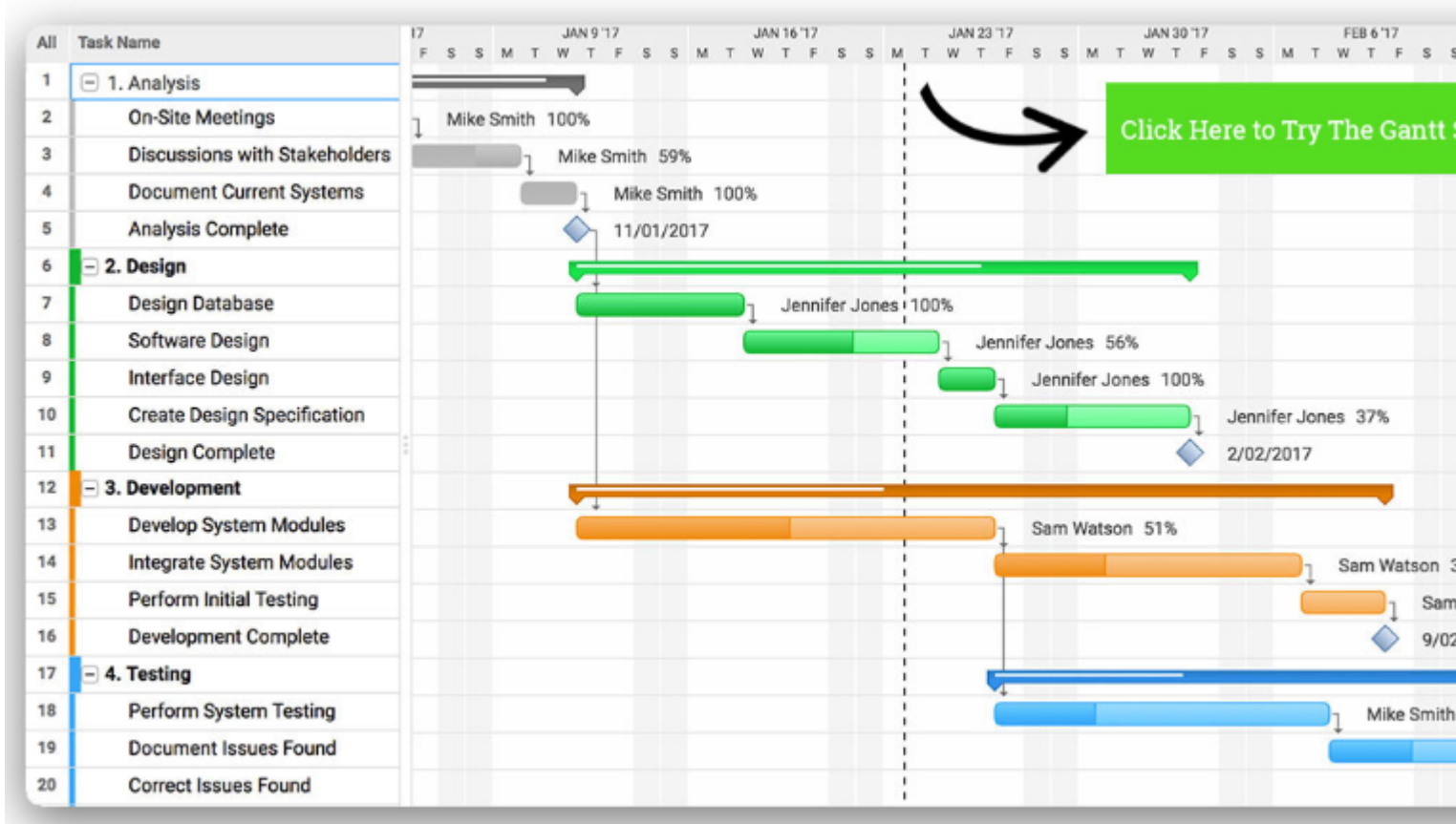
I already have a tree and an array (synced) showing the activities, I will need to sync them with the graph. Also zooming is important, it should be able to zoom in to days (S, M, T, W...) as in the image, and while zooming out it could show weeks, then months, quarter and finally year.

I won't really need layout designer, in any case I'll have to manipulate it with code.

Thanks a lot, I'll try it once you upload it

File Attachments

1) [gantt chart.png](#), downloaded 1599 times



Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
 Posted by [Didier](#) on Fri, 01 May 2020 10:21:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

I moved the discussion to a new Topic since this one is about ScatterCtrl:

[https:// www.ultimatepp.org/forums/index.php?t=msg&goto=53791&#msg_53791](https://www.ultimatepp.org/forums/index.php?t=msg&goto=53791&#msg_53791)

Subject: Re: Ideas: Grid lines at "round" dates or values. More options for tooltip.
 Posted by [Pradip](#) on Fri, 01 May 2020 10:57:07 GMT
[View Forum Message](#) <> [Reply to Message](#)

That's better