technische universität
dortmund

Einführung in die Programmierung

Wintersemester 2009/10

Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering

Fakultät für Informatik

TU Dortmund

Inhalt

- Was ist eine GUI? Was ist QT?
- Erste Schritte: „Hello World!“
- Signals & Slots: SpinBoxSlider
- Anwendung: Temperaturumrechnung
  - Lösung ohne GUI (Ein- und Ausgabe an Konsole)
  - Lösung mit GUI
- Größere Anwendung: Grafik

  zuvor: Qt compilieren und linken

Qt compilieren und linken (I)

Auszug aus Verzeichnisstruktur nach Installation von Qt 4.6.1:

```
- Qt
  - 2010.01
    - qt
      - bin
      - include
      - lib
```

Wurzel der Installation
Version (= 4.6.1)
Beginn von Qt
ausführbare Programme
Header-Dateien
Bibliotheken

Dem Compiler muss gesagt werden,

- wo er die Header-Dateien zum Compilieren finden kann:

```
C:\Qt\2010.01\qt\include;C:\Qt\2010.01\qt\include\Qt
```

- wo er die Bibliotheken zum Linken finden kann:

```
C:\Qt\2010.01\qt\lib
```

- welche Bibliotheken er zum Linken verwenden soll: **d** → debug

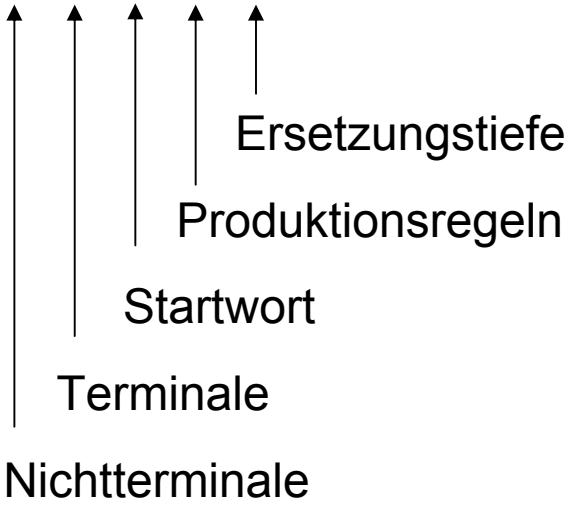
```
QtCore4.lib QtGui4.lib
```
bzw.
```
QtCored4.lib QtGuid4.lib
```

Lindenmayer-Systeme (L-Systeme) nach Aristid Lindenmayer, theoret. Biologe, Ungarn

Intention: axiomatische Theorie zur biologischen Entwicklung

Formalismus: Ersetzungssysteme ähnlich zu formalen Grammatiken

Quintupel: (N, T, ω, P, n)



hier:

< 6

1 Regel: F → ...

beliebig aus N ∪ T

+ – | [ ]

F

Vorgehensweise (gemäß unserer Einschränkungen): Schritt 1

```
setze s = ω (Startwort)
while (n > 0)
  initialisiere leere Variable t
  laufe von links nach rechts über s:
    falls Terminal dann nach t kopieren
    falls Nichtterminal F dann rechte Seite der Produktionsregel nach t kopieren
  setze s = t
  setze n = n – 1
endwhile
```

Bsp: ({ **F** }, { **+, -, [,], |** }, **F+F**, { **F→F--F** }, 2)

F+F → F--F+F--F → F--F--F--F+F--F--F--F

Vorgehensweise (gemäß unserer Einschränkungen): Schritt 2

sei s das erhaltene Wort nach n Ersetzungsrunden

setze (x_0, y_0, α_0) als Startwert fest, setze k = 0, λ = Schrittweite, β = Winkel

laufe über s von links nach rechts

falls F: $(x_{k+1}, y_{k+1}, \alpha_{k+1}) = (x_k + \lambda \cos \alpha_k, y_k + \lambda \sin \alpha_k, \alpha_k)$;
zeichne Linie von (x_k,y_k) nach (x_{k+1},y_{k+1})

falls +: $(x_{k+1}, y_{k+1}, \alpha_{k+1}) = (x_k, y_k, \alpha_k + \beta)$;

falls –: $(x_{k+1}, y_{k+1}, \alpha_{k+1}) = (x_k, y_k, \alpha_k - \beta)$;

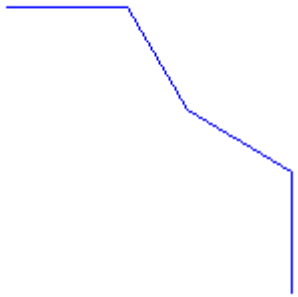
falls | : $(x_{k+1}, y_{k+1}, \alpha_{k+1}) = (x_k, y_k, \alpha_k - 180°)$;

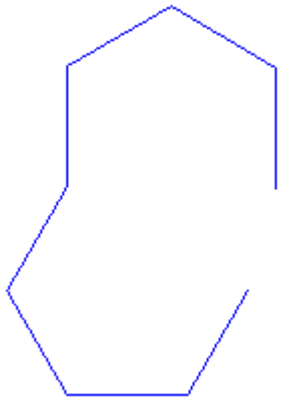
falls [: **push** (x_k, y_k, α_k); $(x_{k+1}, y_{k+1}, \alpha_{k+1}) = (x_k, y_k, \alpha_k)$;

falls] : $(x_{k+1}, y_{k+1}, \alpha_{k+1})$ = `top(); pop()`

setze k = k + 1

Bsp: **F+F** → **F--F+F--F** → **F--F--F--F+F--F--F--F** $\beta = 30°$





... noch nicht spektakulär ...

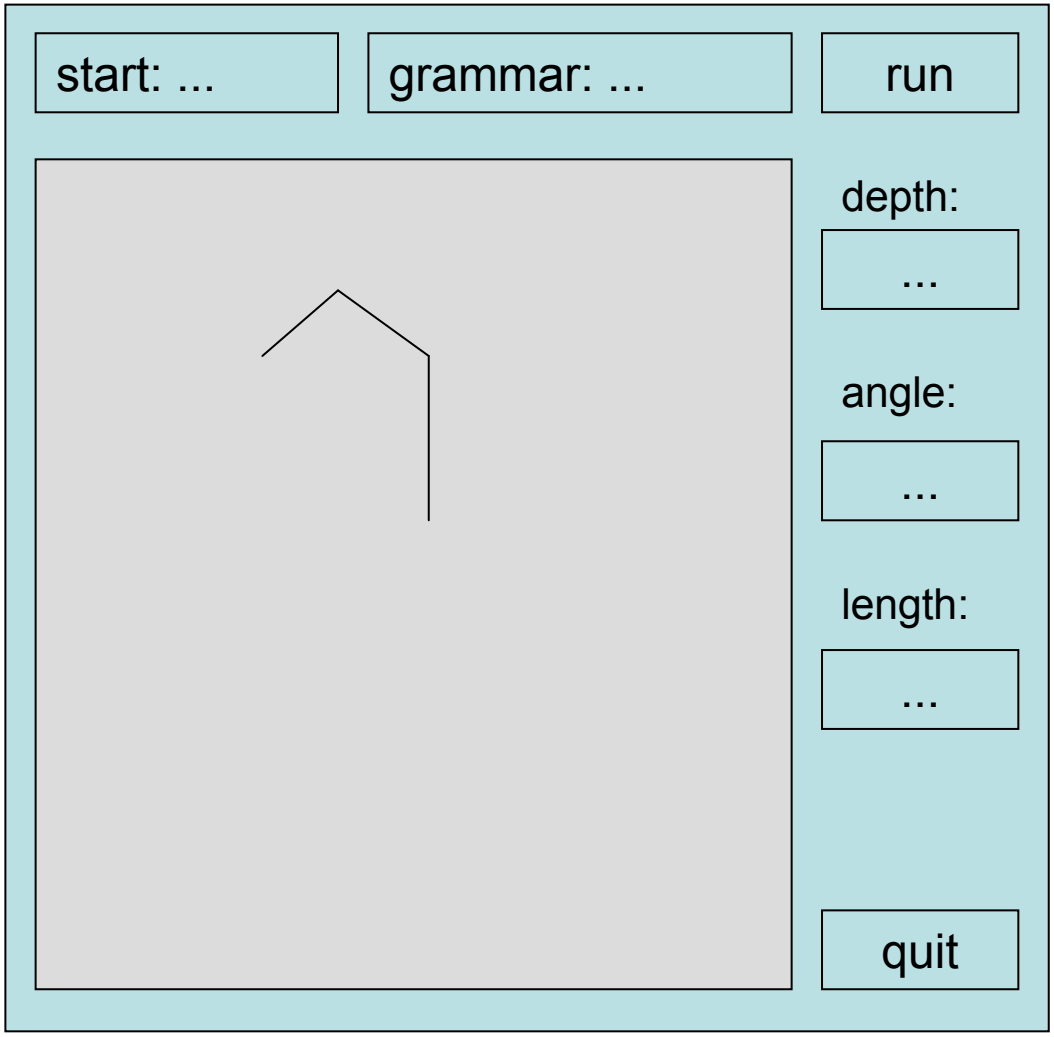
Planung der GUI

`class Canvas : public QWidget`

sorgt für die Darstellung eines L-Systems

`class Window : public QWidget`

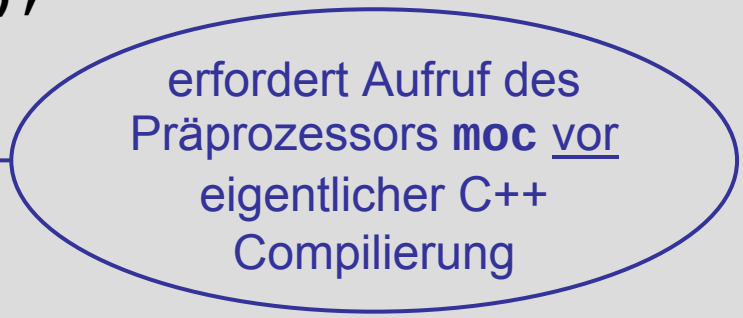
verwaltet alle Controls



```
class Window : public QWidget {
Q_OBJECT
public:
  Window(QApplication *aApp);
  ~Window();

public slots:
  void run();

protected:
  QApplication *fApp;
  QLineEdit *fStart, *fGrammar, *fLength;
  QPushButton *fRun, *fQuit;
  QSpinBox *fDepth, *fAngle;
  QLabel *fLabelStart, *fLabelGrammar, *fLabelLength,
         *fLabelDepth, *fLabelAngle;
  QSlider *fSliderH, *fSliderV;
  Canvas *fCanvas;
};
```



Datei

Window.h

Auszug aus Verzeichnisstruktur nach Installation von Qt 4.6.1:

```
- Qt
  - 2010.01
    - qt
      - bin
      - include
      - lib
```

Wurzel der Installation
Version (= 4.6.1)
Beginn von Qt
ausführbare Programme
Header-Dateien
Bibliotheken

Aufruf des Prä-Compilers **moc** vor eigentlicher C++ Compilation:

→ als *pre-build event* oder ähnliches eintragen bzw. explizit aufrufen:

```
C:\Qt\2010.01\qt\bin\moc -o WindowMeta.cpp Window.h
```

Datei, die erzeugt wird

Datei, die **slot** enthält

```
class Canvas : public QWidget {
Q_OBJECT

public:
  Canvas(QWidget *aParent = 0);
  void draw(QString &aStart, QString &aGrammar, int aDepth,
       QString &aLength, int aAngle, int aPosX, int aPosY);

protected:
  void paintEvent(QPaintEvent *aEvent);  // überschrieben

private:
  QString fStart, fGrammar;
  int fDepth, fLength, fAngle;
  QPoint fStartPos;
  QRectF exec(QString &aRule, QPainter *aPainter);
};
```

Implementierung der Klassen

⇒ live demo ... (mit MS Visual Studio 2008)

Demo mit Beispielen

```
start: F                grammar: F → F[-F]F[+F][F]
degrees 20      length 5        depth 5

start: F-F-F-F grammar: F+F-F-FF+F+F-F
degrees 90      length 5        depth 5

start: F-F-F-F-F-F      grammar: F+F--F+F
degrees 60      length 5        depth 4

start: F                grammar: FF-[-F+F+F]+[+F-F-F]
degrees 20      length 4        depth 4

start: F                grammar: F[+F]F[-F]F
degrees 20      length 4        depth 4
```