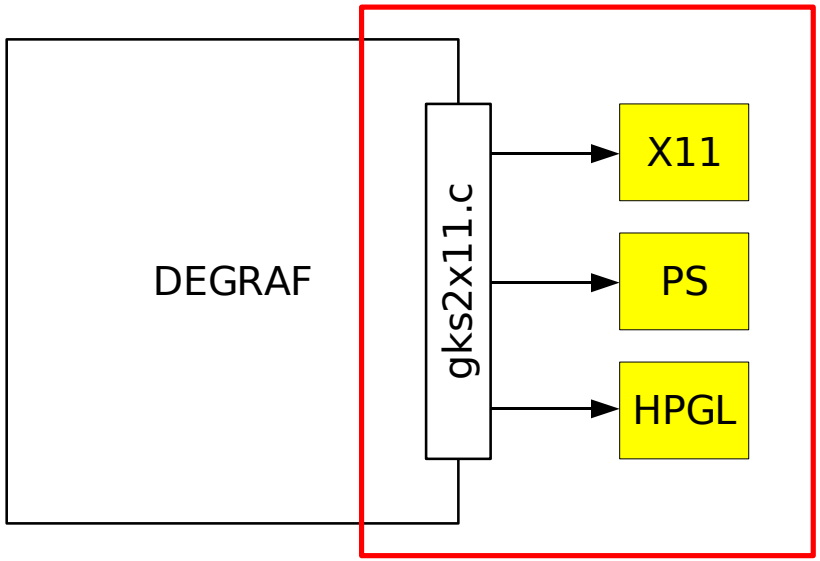
# Entwurf und Implementierung einer generischen Backend-Infrastruktur für den Prozessimulator DUPSIM

Christiane Berndt

25. Februar 2010

# Altes System




Das System ist derzeit:

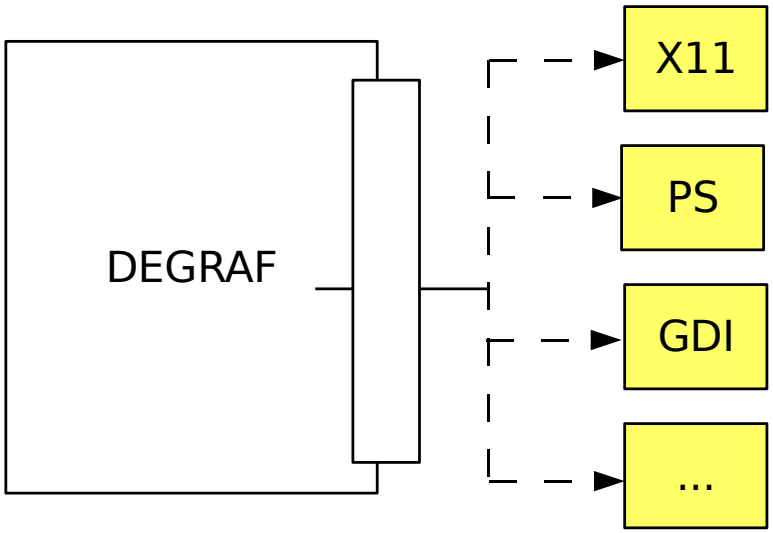
- monolithisch,
- unflexibel und
- plattformabhängig.

# Neues System

Anforderungen an das neue Backend:

- Realisierung der einzelnen Ausgabegeräte in eigenen Klassen
- Flexible Anbindung eines oder mehrerer Ausgabemedien
- Parallele Ausgabe auf mehrere Geräte
- Einfachere Erstellung neuer Ausgabeformate
- Implementierung der Ausgabe in ein Windows-Fenster

## Neues System




Benötigt werden:

- Grafikbibliothek,
- Programmiersprache und
- Klassenstruktur.

# Grafikbibliothek

Betrachtete Grafikbibliotheken:

- PostScript
- X11 und Xlib
- Java2D
- GDI
- Cairo

# Vergleich der Grafikbibliotheken

| PS | X11 | Java2D | GDI | Cairo |
|---|---|---|---|---|
| *Plattform* | | | | |
| Windows, Unix & Derivate | Windows, Unix & Derivate | alle Desktop-Systeme | Windows | alle Desktop-Systeme |
| *Koordinatensystem* | | | | |
| positiv | negativ | negativ | negativ | negativ |
| *Bilder & Text* | | | | |
| ja | ja | ja | ja | ja |

| PS | X11 | Java2D | GDI | Cairo |
|---|---|---|---|---|
| *Farben* | | | | |
| Graustufen, RGB, CMYK | Graustufen, RGB, S/W | RGB | RGB | RGB |
| *Affine Transformationen* | | | | |
| ja | ja | ja | ja | ja |
| *Lizenz* | | | | |
| - | MIT | GNU Classpath | - | LGPL |

# Programmiersprachen

## Fortran

+ Implementierungsspra-che von DUPSIM und DEGRAF

- Kein Xlib Interface für Fortran77
- Keine Klassenstruktur

## C

+ C-Code von Fortran aus aufrufbar

- Keine Klassenstruktur

## Java

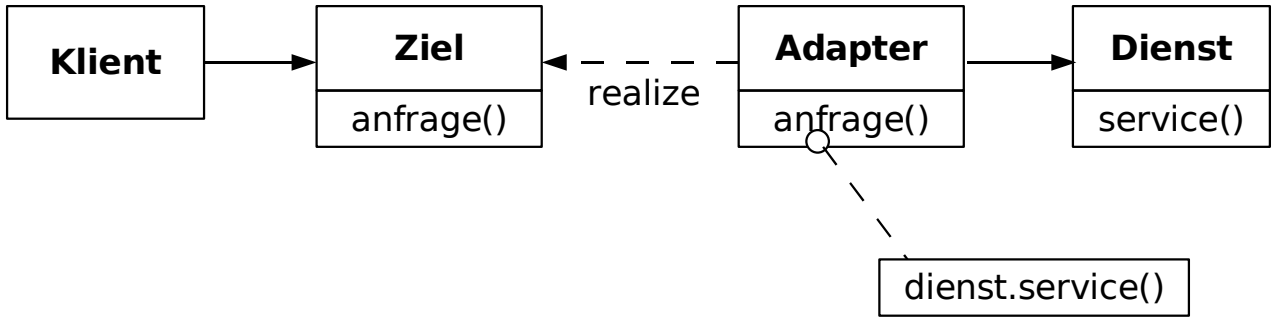
+ Klassenstruktur
- Verwendung des JNI und der JVM

## C++

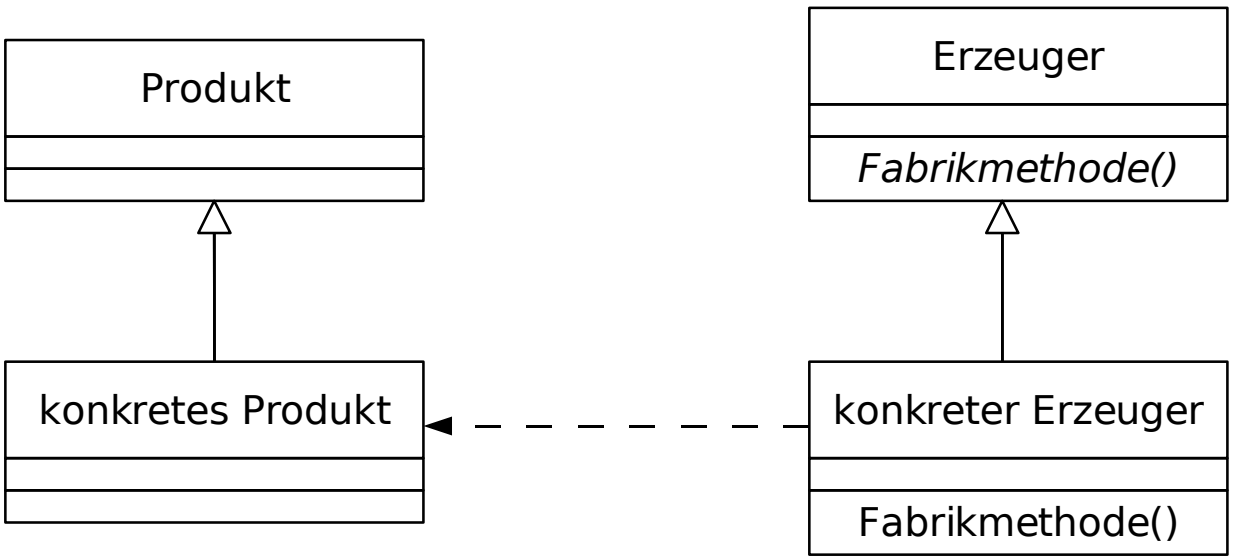
+ Klassenstruktur
+ C als Untermenge

# Entwurfsmuster

- Adapter:
  - Übersetzung zwischen inkompatiblen Schnittstellen
  - Schnittstelle zwischen Fortran und C++-Code



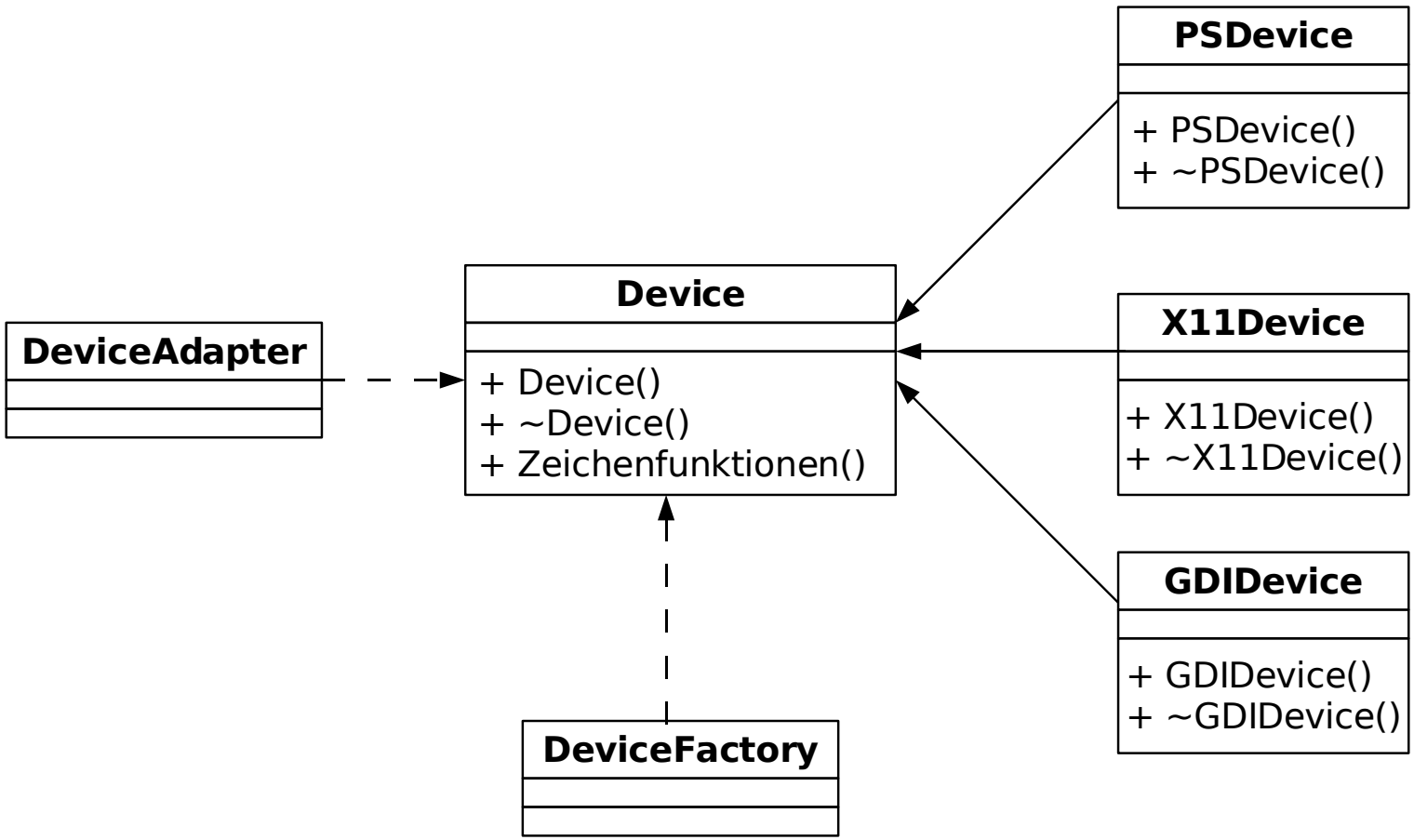
- Factory:
  - Erzeugung eines Objektes, das durch eine Unterklasse spezifiziert wird
  - Generieren einzelner Devices



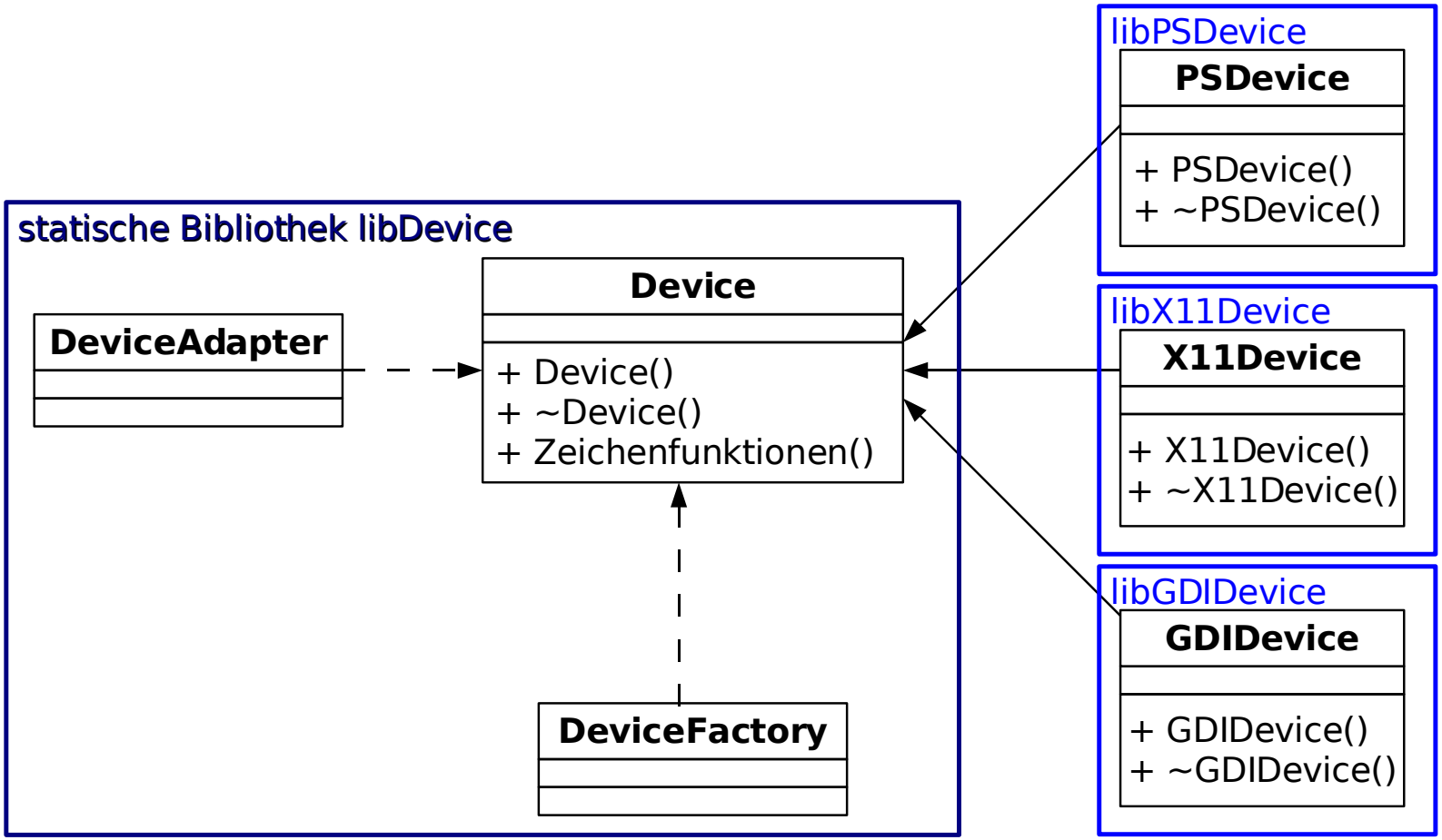
# Klassen

- Adapter: Schnittstelle zwischen Fortran und C++-Code
- Factory: Erzeugung einzelner Devices
- Allgemeine Device-Klasse: Realisierung aller Grafikfunktionen, die durch Cairo für alle Ausgabemedien gleich sind
- Einzelne, plattformspezifische Devices

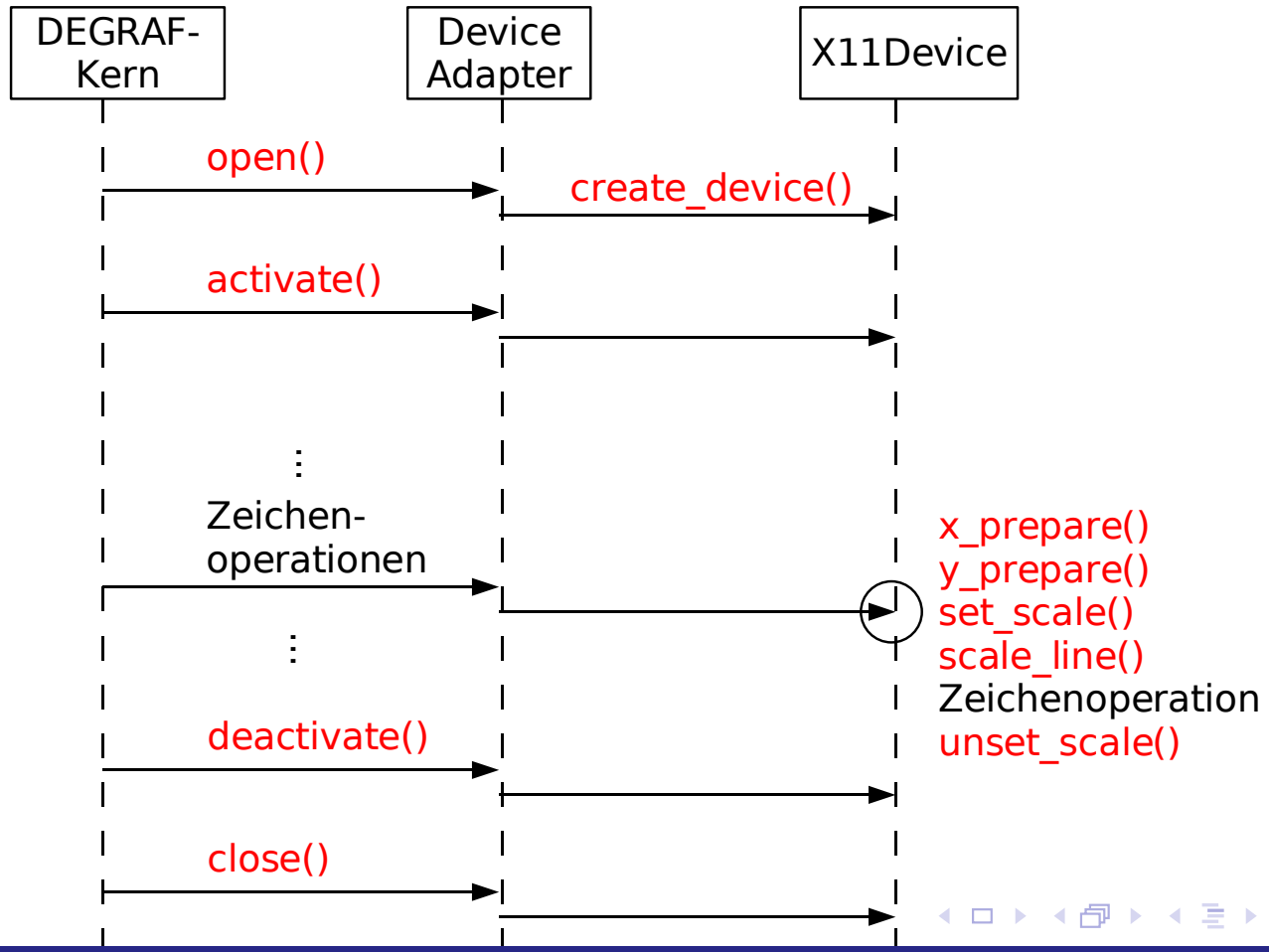
# Klassenstruktur



# Klassenstruktur



# Programmablauf - Ein Beispiel



# Erstellung eines neuen Device

- Zu implementierende Funktionen:
  - Factory-Methode $Device * get\_device()$
  - Öffnen, Schließens, Aktivieren, Deaktivieren und Bereinigen des Ausgabegerätes
  - 8 weitere Funktionen, die von Koordinatentransformationen abhängig sind
- Eintrag einer ID in eine Textdatei, die IDs auf Bibliotheken abbildet
- Kompilieren in eine dynamische Bibliothek

# Loggen und Redraw

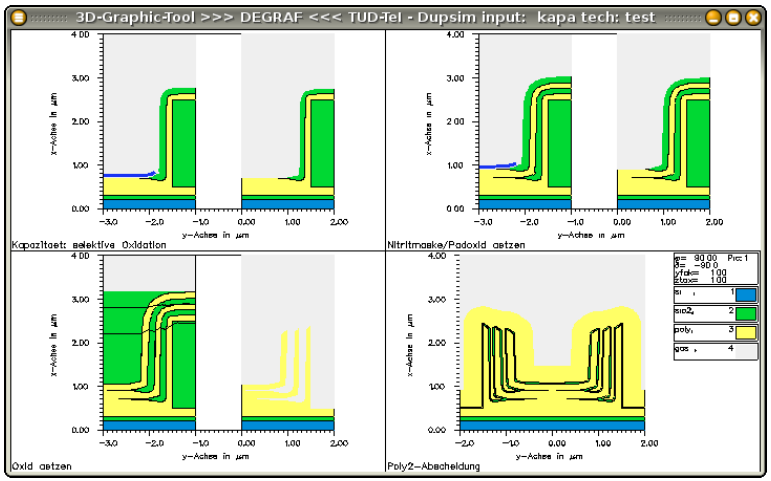
- Ziel: Anpassen des Fensterinhaltes an veränderte Fenstergröße
- Speichern der ausgeführten Grafikfunktionen in eine Liste
- Thread zum Empfangen von Tastatur- und Mausereignissen
- Bei Erhalt eines Ereignisses:
  - Skalierungsfaktor berechnen und setzen (*cairo_set_scale*())
  - Setzen der neuen Fenstergröße
  - Erneutes Ausführen der geloggten Grafikfunktionen (*redraw*())

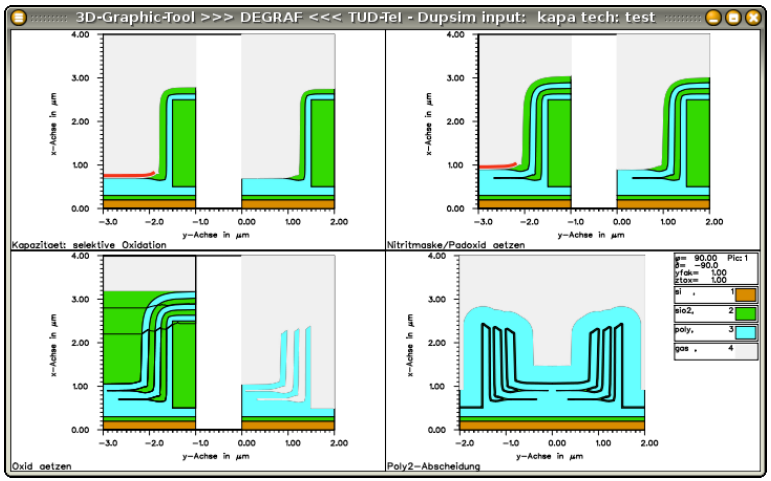
# Vergleich der Ausgaben

- Im Original-DEGRAF: Farbunterschiede zwischen X11 und PS
- Jetzt: einheitliche Farbgebung
- Keine weiteren Unterschiede zwischen originaler und neuer Ausgabe

## Vergleich der Ausgaben

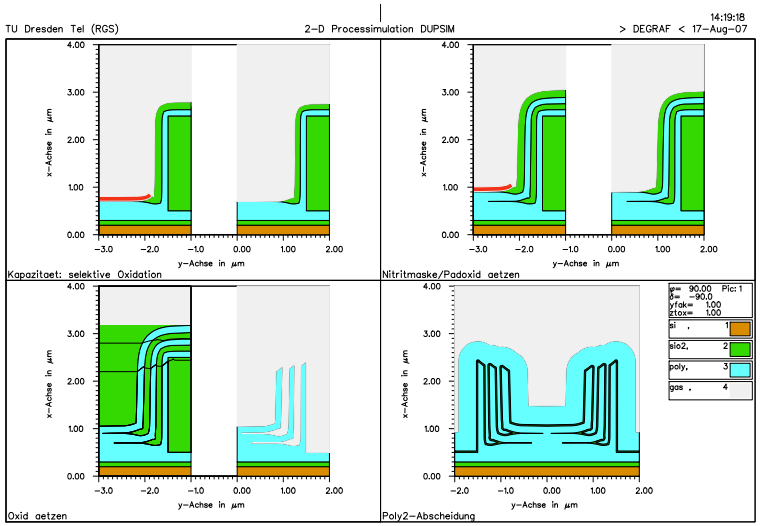
# Ausgabe des alten und des neuen DEGRAF nach X11:

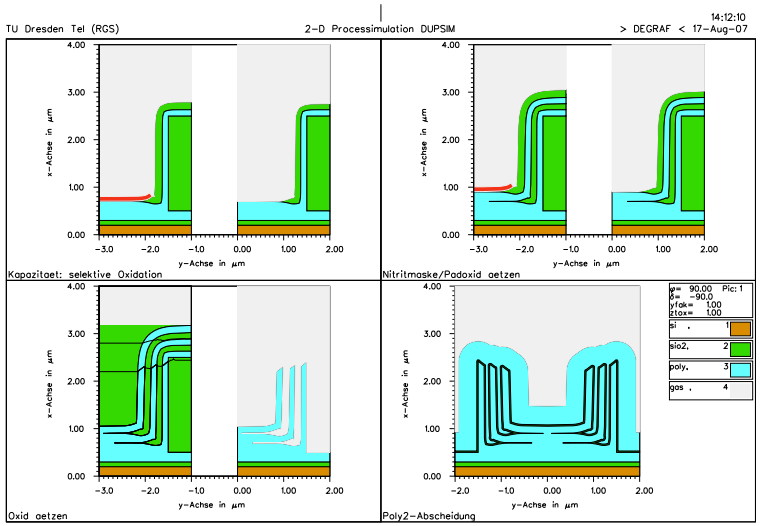


## Vergleich der Ausgaben

# Ausgabe des alten und des neuen DEGRAF nach PostScript:



# Fazit

- Keine Veränderungen in der Bedienung von DEGRAF
- Über Device-Grenzen einheitliches Farbschema
- Komfortablere X11-Ausgabe
- Nicht-monolithisches Backend durch Verwendung einer Klassenstruktur
- Flexibilität durch Nutzung dynamischer Bibliotheken
- Einfache Erweiterbarkeit

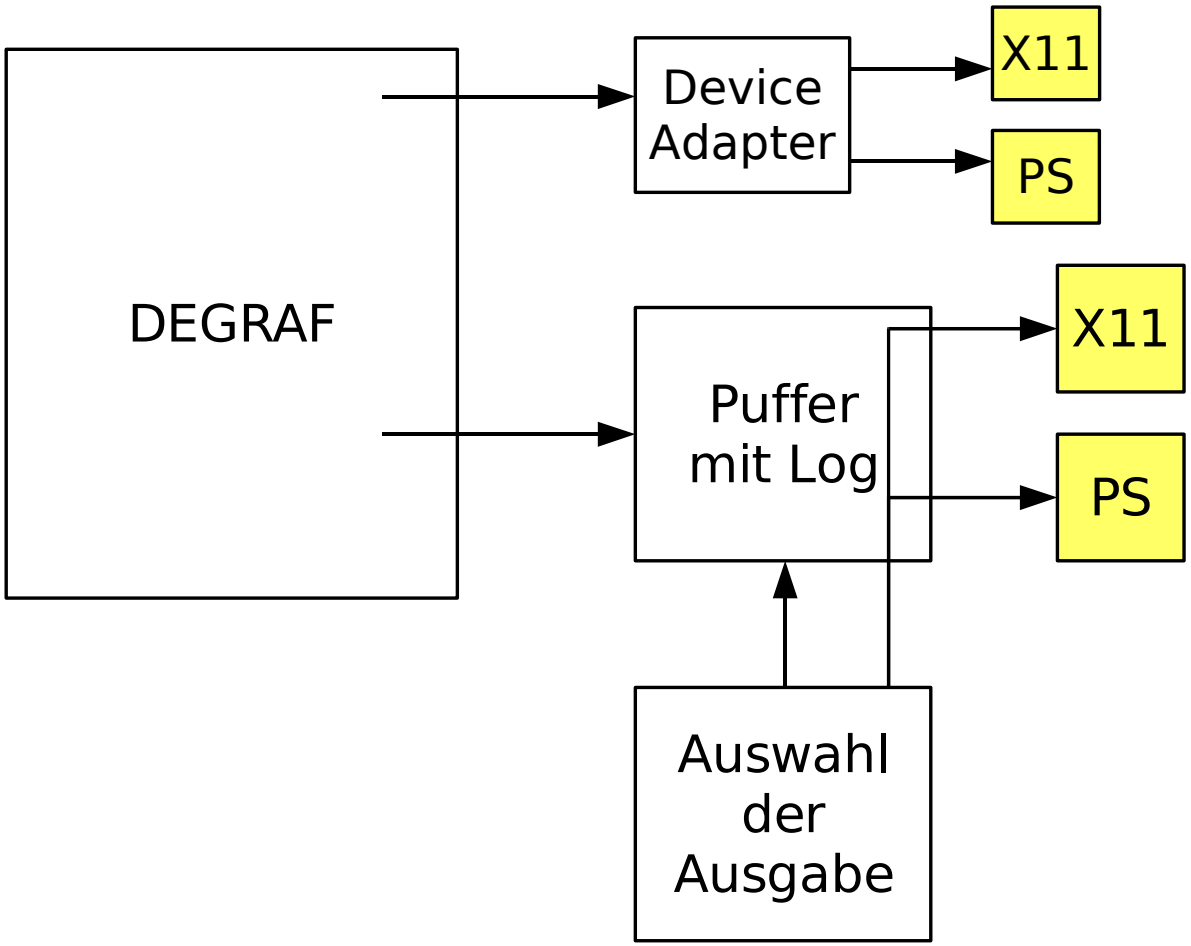
# Demo

...

# Zusammenfassung und Ausblick

- Zusammenfassung:
  - Die am Anfang definierten Ziele wurden erreicht.
  - Es erfolgten wesentliche Verbesserungen von DEGRAF.
- Ausblick:
  - Mehr Flexibilität durch Veränderungen am DEGRAF-Kern
    - Stärkere Entkopplung von Kern und Backend
    - Realisierung einer flexiblen Auswahl des Ausgabeformates
  - Realisierung weiterer Ausgabemedien (z.B. SVG)

# Ausblick


DEGRAF
Device Adapter
X11
PS
Puffer mit Log
X11
PS
Auswahl der Ausgabe

Vielen Dank für Ihre Aufmerksamkeit.
Gibt es noch Fragen?