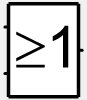
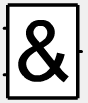
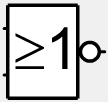
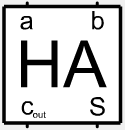
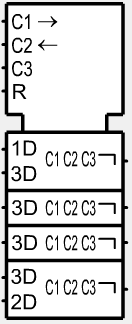
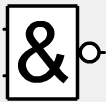
# Logik Simulator



Projektteam: Moritz Wahlenmeier, Bastian Schäfer &   
. Stephen Heisser

Auftraggeber: Markus Greiner



# Inhaltsverzeichnis

* [Logik Simulator](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554222) 1
* [Inhaltsverzeichnis](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554223) 2
* [Allgemeine Informationen](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554224) 3
* [Projektbeschreibung](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554225) 4
* [Funktionsweise (kurz)](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554226) 5
* [Optimierungsmöglichkeiten](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554227) 6
* [Funktionsweise (Lang)](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554228) 7
* [Zustand](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554229) 10
* [Wie es vorgefunden wurde:](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554230) 10
* [Momentaner Zustand:](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554231) 11
* [Zeitplan](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554232) 12
* [Geplantes Material](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554233) 12
* [Probleme](../../../../Mar3t/Documents/GitHub/Doku_me.docx#_Toc10554234) 13

# Allgemeine Informationen

**Team:**

Moritz Wahlenmeier   
 moritz.wahlenmeier@yahoo.de

Bastian Schäfer  
 schaefer2001@web.de

Stephen Heisser  
 st.heisser@gmail.com

**Auftraggeber:**

Herr Markus Greiner  
greiner@gds2.de  
Lehrer Gottlieb Daimler Schule 2  
Böblinger Str. 73, 71065 Sindelfingen

# Projektbeschreibung

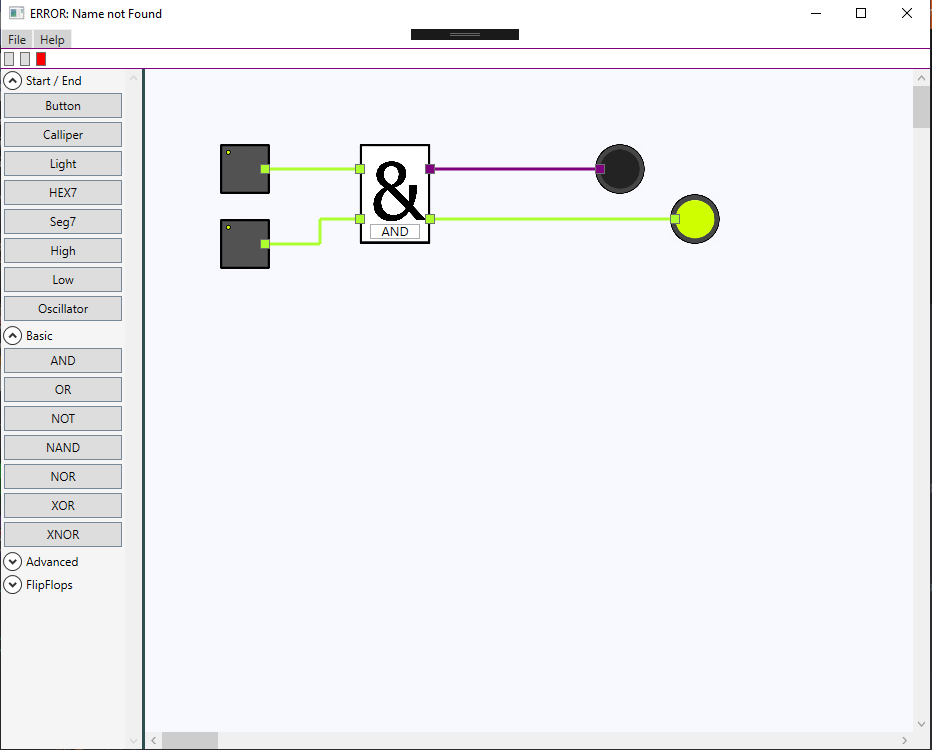
Das Ziel ist das ablösen des alten, auf Flash basierendem, Programms „Logiflash“, durch eine auf Windows Presentation Foundation (WPF) basierende Desktopanwendung.   
Da „Logiflash“ alte Symbole beinhaltet und in vielen Situationen nicht mehr von Browsern unterstützt wird. Des Weiteren ist das Programm aktuell nicht sehr benutzerfreundlich. „Logiflash“ wird aktuell im Praktikum Informatik Unterricht im ersten Schuljahr des 2BKI benutzt und soll gegebenenfalls durch das neue Programm abgelöst werden.

Hierzu muss das Simulationsprogramms zum Erstellen und Testen von logischen Schaltungen erstellt werden.

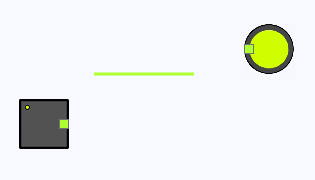
Als Unterstützung dazu, das Projekt an mehreren Orten zu bearbeiten, wird das Programm „Github Desktop“ verwendet.

# Funktionsweise (kurz)

In der unteren rechten Leiste befindet sich eine Liste mit allen Logik Gattern als auch Anfangs- und Endbausteine wie Buttons und Lampen, welche per Drag und Drop auf die Arbeitsfläche gezogen werden können. Um diese zu verbinden klickt man auf die Output Fläche (Quadrate rechts an Baustein) und zieht ein Kabel auf die Input Fläche (Quadrate links an Baustein). Kabel können nur vertikal oder horizontal gezogen werden. Auf jedes vertikale folgt ein horizontales Kabel und andersrum. Schaltungen können unter dem Menüpunkt File oder durch Shortcuts in der zweiten Zeile gespeichert, geöffnet oder gedruckt werden. Durch drücken des roten Icons in der zweiten Zeile wird der Lösch-Modus aktiviert oder deaktiviert. Im Lösch-Modus werden Gatter und Kabel durch Anklicken gelöscht.

  
logisch 0: lila logisch1: grün

# Optimierungsmöglichkeiten

****

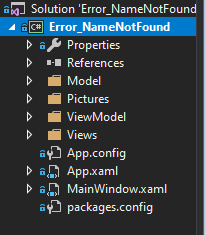
* Beim Verschieben der Bausteine bleiben verbundene Kabel an ihrer vorherigen Position die logische Verbindung bleibt aber weiterhin bestehen.

Optimiert: Algorithmus setzt Kabelende /Anfang an neue Position und baut je nach Verschiebung benötigte vertikale/horizontale Kabel ein (Aufwendig).

* Manche bereits existierende Bilder müssen noch angepasst werden.

# Projekt Baum

Das Programm ist in Model (Logik), Pictures (Bilder), ViewModel() und Views() aufgeteilt. Zudem gibt es noch das MainWindow (Programmfenster).



Model wiederum besteht aus Gates (Logikgatter), dort sind alle 81 Logikgatter und die Basisklasse enthalten.  
Basemodel und RelayCommand sind beide für live Änderungen an der Oberfläche zuständig und Basisklassen der Basisklasse der Logikgatter und der Funktionen Speicher, Laden, Drucken & Hilfe.

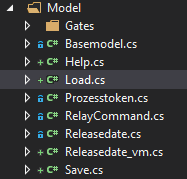
Der Prozesstoken ist dafür zuständig, dass wenn ein Bearbeitungszyklus gestartet wird, dass dieser alle dazugehörigen Verbindungen durchgeht und dann sich zum Schluss auch wieder beendet.

Save ist für das Speicher, der Logikgatter, zuständig.

Load ist für das Laden, der Logikgatter, zuständig.

Help ist für die Hilfeanzeige zuständig.

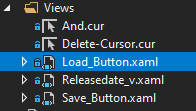
Releasedate(\_vm) sind für das Easteregg zuständig, das anzeigt wie lange das Programm bereits fertig sein sollte.



Pictures beinhaltet alle Bilder, die in der Oberfläche benutzt werden.

ViewModel beinhaltet alle XAML Dateien für die Logikgatter, wie diese aussehen und sich verhalten.

Views beinhaltet die XAML Dateien für die Load, Save und das Easteregg.



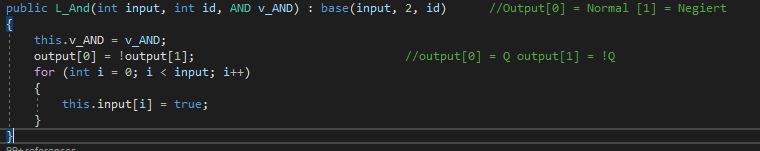
# Funktionsweise (Lang)

* **Logikgatter**:

Alle Logikgatter müssen soweit verstanden sein, dass man sie mit C#  
 programmieren kann. Dabei muss darauf geachtet werden, dass manche Logikgatter auch mit unterschiedlich vielen Eingängen funktionieren müssen.

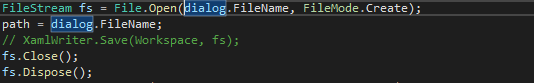
z.B.:

Ein UND/AND Gatter kann von 2 bis 8 Eingänge haben.

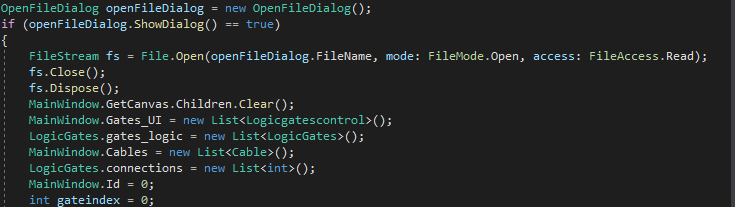


* **Speichern/Laden**

Beim Speichern (hier mit FileStream) müssen alle Listen ausgelesen werden und dann in einer Datei gespeichert werden.

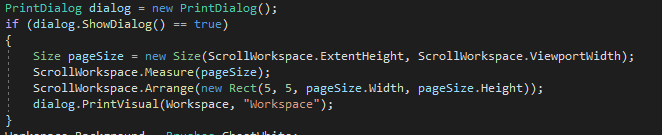


Beim Laden (was hier mit Filestream erledigt wird) müssen alle Listen zuerst geleert werden, damit die alten Gatter nicht die neuen Gatter beeinflussen.



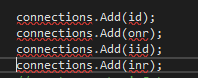
* **Drucken**

Der Canvas muss vom Programm ausgelesen werden und dann ausgedruckt werden.



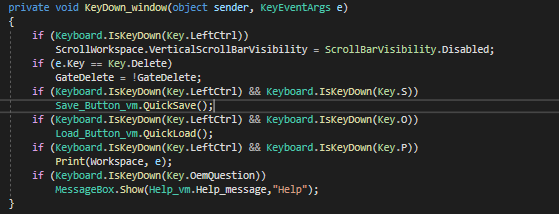
* **Verbindungen**:

Verbindungen arbeiten mit der einzigartigen Id jedes Logikgatters und jeder Verbindungslinie. Wird eine Verbindung eingeleitet, so wird in einer Liste die Id des Ausgangsgatts gespeichert, welcher Ausgang, die Id des Eingangsgatters und welchen Eingang die Verbindung hat.



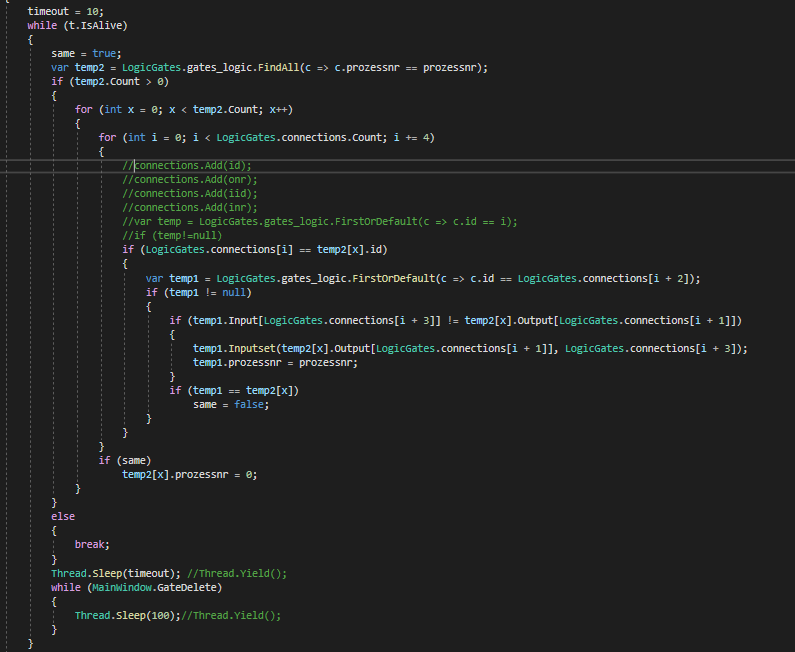
* **Tasten-Kürzel**

Hier musste ein Tasten druck abgefragt werden solange das Programm sich im Focus befindet. Falls dies der Fall war wurde die dazugehörige Funktion aufgerufen.



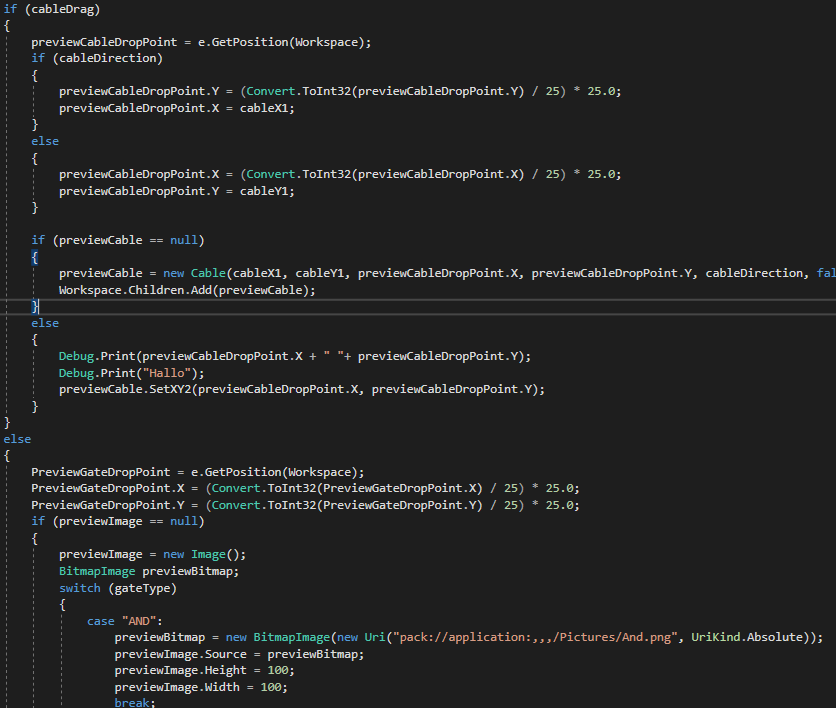
* **Prozesstoken**:

Wenn ein Taster oder ähnliches ein Signal abgibt wird ein „Prozesstoken“ erstellt der innerhalb eines neuen Threads die Liste mit den Verbindungslinien durchgeht und überprüft ob diese durch das Signal verändert wird. Falls nichts mehr verändert wird, wird der Thread beendet.

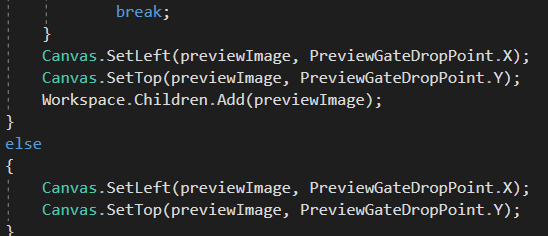


* Drag Drop

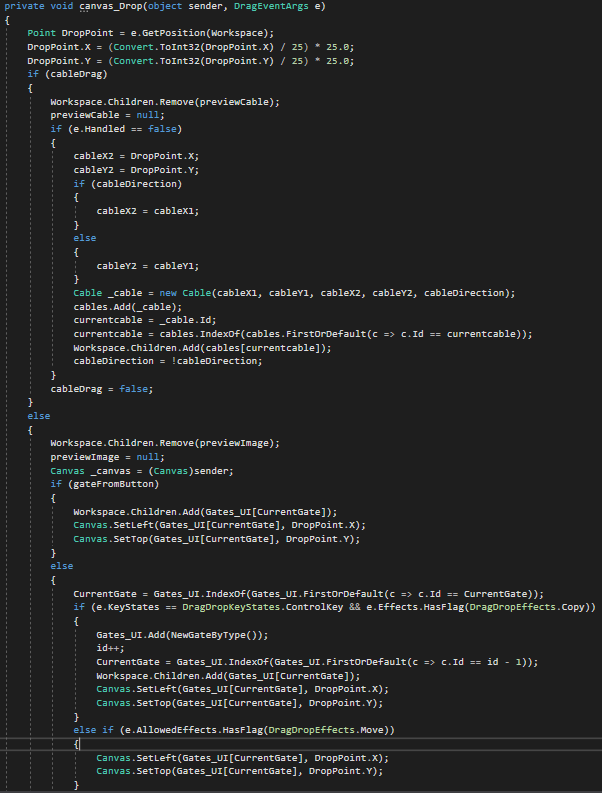
Wird ein Baustein oder Kabel über den Canvas gezogen wird bis zum Drop eine Preview generiert, dessen Position auf dem Canvas wiederholend aktualisiert wird, die zeigt wo das Element beim Drop positioniert wird



(restliche cases aus platz gründen ausgeschnitten )

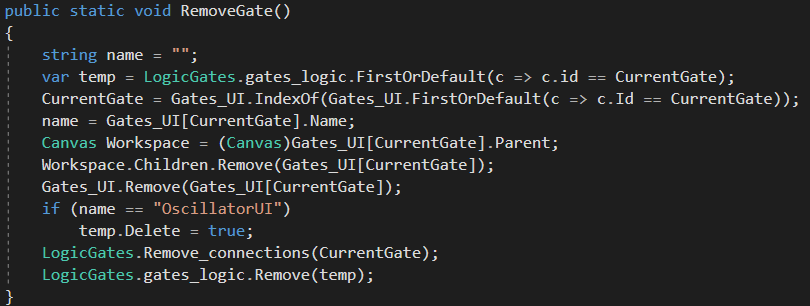
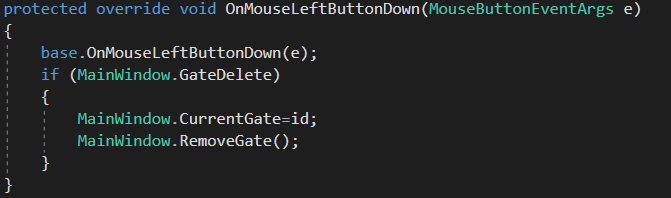


Beim Drop wird die Preview gelöscht und durch ein, je nachdem ob Steuerung gedrückt wird, neues oder bisher vorhandenes Element ersetzt



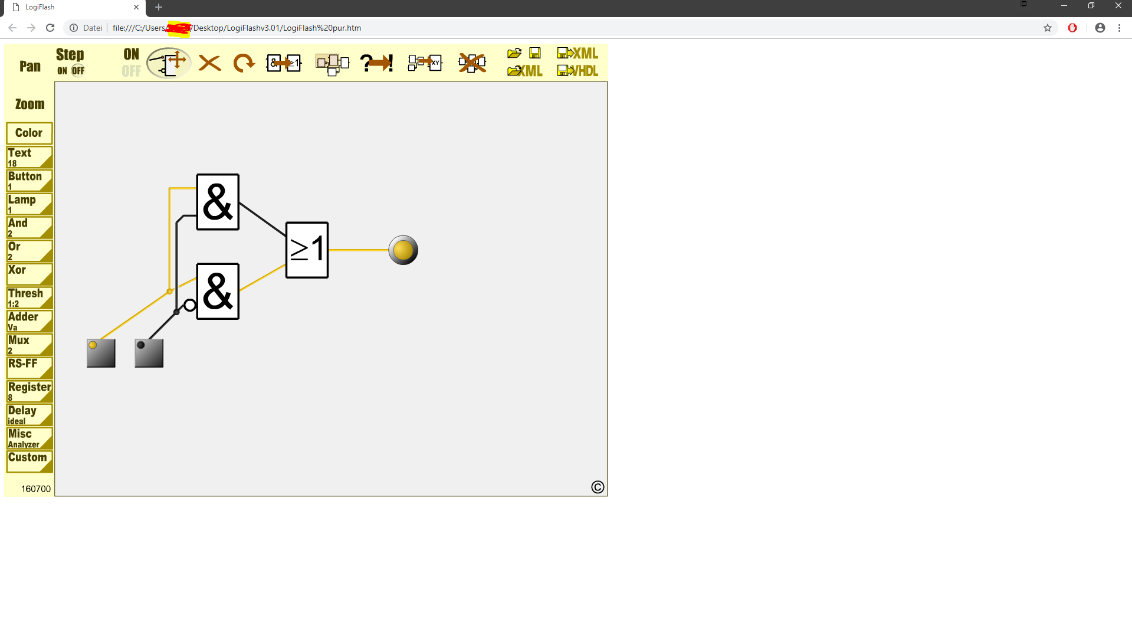
* Löschen

Wird ein Element während der Lösch-Modus aktiviert ist angeklickt wird dieses aus der Gatter Liste entfernt und die logischen Verbindungen gelöscht



# Zustand

## Wie es vorgefunden wurde:



Button1

Button2

* Nicht skalierend abhängig von Bildschirmgröße
* Linien ziehen nicht anwenderfreundlich
* Testfunktion änderbar im laufenden Betrieb
* Flash basierend (wird häufig nicht mehr von Browsern unterstützt)
* Veraltete Symbole
* Umständliches umstellen für Löschen von Bausteinen
* Text ist nicht an Bausteine gebunden und keine Pflicht
* Benötigt ein weiteres Programm, um Schaltungen auszudrucken

## Momentaner Zustand:

* Erreicht:
* Skalierend
* auf (WPF) basierende Desktopanwendung
* Tasten-kürzel
* Direktes Drucken aus dem Programm heraus
* Feste namens Zuordnung zu Bauteilen
* Nicht erreicht:

- Debugging Umgebung

- Kontextmenü

- Möglichkeit simple Programme auf einem Mikrokontroller zu testen

# Zeitplan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Wahlenmeier** | Zeit in h | **Schäfer** | Zeit in h | **Heisser** | Zeit in h |
| Programm Organisation | 2 | Grafikdesign | 50 | Logik Bausteine in C# | 29 |
| Programm in C für Mikrokontroller | 0 | Tastatur Kürzel | 0 | Verbindungslinien | 32 |
| Schnittstelle für Mikrokotroller zu C# | 0 | Englische und deutsche Bedienelemente | 0 | Abspeichern und öffnen von Programm | 28 |
| Drag & Drop | 54 |  |  | Fleißarbeit | 35 |
| Löschen / zoomen /  Toolbox | 8 |  |  | Tastatur Kürzel | 5 |
| Fehlerdiagnose | 24 | Fehlerdiagnose | 11 | Fehlerdiagnose/Cleanup | 53 |
| Lern Zeit | 26 | Lern Zeit | 10 | Lern Zeit | 54 |
| PowerPoint | 2 | PowerPoint | 2 | PowerPoint | 2 |
|  |  |  |  |  |  |
| Gesamtzeit | 117 | Gesamtzeit | 73 | Gesamtzeit | 238 |

# Geplantes Material

|  |  |  |  |
| --- | --- | --- | --- |
| **Materialname** | **Anzahl** | **Preis in Euro €** | **Gestellt**  **von Team** |
| XC886 Mikrokontroller | 1 | 60 | X |
| USB-A zu USB-B Kabel | 1 | 1 | X |
|  |  |  |  |
| Gesamtpreis |  | 61 |  |

# Probleme

* Bugs
  + Neue Funktionen haben häufig alte Funktionen behindert
  + CPU Überlastung bei der zyklischen Abarbeitung des Programms
  + Bildschirm Größen unterschiede
  + Schul-PCs zu langsam
* Keine richtige Arbeitseinteilung