Rückblick

Über's Semester viele Ziele erreicht!

- kein überfüllter Hörsaal mehr
- bessere Betreuung da sinnvolle Kontaktzeiten erhöht
  (= aufgewendete Zeit / Anzahl erfolgreicher Studierender)
- Betreuungskosten (= aufgewendete Zeit) nicht erhöht
- Erfolge dauerhaft, geringere Belastung späterer LVAen
- mehr Zeit für Lernen, weniger für Sarkasmus, Polemik, ...

Wie jedes Semester steht die Drohung im Raum:

- „Im nächsten Semester machen wir es noch besser!“

Was auffällt

- Großteil kann mit komplizierten Themen gut umgehen
- einige typische Fehler sind sehr schwer zu beseitigen
- mit viel Mühe kann man es trotzdem schaffen
- auch erfahrene Leute wissen und können nicht alles
- Belastbarkeitsgrenze viel höher als angenommen
- gelegentlich Konzentration auf andere LVAen?

Ausblick

PK geschafft! Was nun?

- Algorithmen und Datenstrukturen
- Objektorientierte Modellierung
- Programmierparadigmen
  (Objektor. Prog.techniken, Funktionale Programmierung)
- Software Engineering
- ...

Kann ich jetzt Programmieren?

- in PK nur Grundlagen vermittelt
- Fähigkeiten entwickeln sich langsam mit der Praxis
- Programmieren lernt man in anderen LVAen nicht, aber viele LVAen entwickeln Programmierfähigkeiten weiter
- Erfahrungen in einer Sprache lassen sich leicht auf andere Sprachen übertragen
- man hat nie ausgelernt

PK nicht geschafft – was nun?

- Ursache ermitteln
  - Anfang verschlafen oder Aufwand unterschätzt
  - zeitliche, organisatorische, inhaltliche Probleme
  - „Will ich wirklich Informatik machen?“
  - Ausrede „Pech gehabt“ gilt nicht
- Ursache vermeiden (oder es bleiben lassen)
- zweite Chance nutzen, jetzt aber wirklich
- eine dritte Chance wird es für viele nicht geben
  (formal schon, aber durch STEOP stark erschwert)

Neuerungen in PK

- Sommersemester: voraussichtlich von *Anton Ertl* gehalten
- Übungen weiter von *Karl Gmeiner* betreut
- sicher einige Änderungen im Stoff und in der Organisation
- Programmierpraxis geht wie gewohnt weiter

Vorschläge für die Ferien

Sudokus lösen

- Motivation: Gehirntraining immer gut
- Vorgehensweise:
  - Regeln lernen
  - beispielhaft einige Sudokus lösen
  - dabei einen Lösungsansatz entwickeln (Abstraktion)
  - Lösungsansatz in Programm umsetzen
  - durch Automatisierung freigesetzte Kapazitäten sinnvoll nutzen, beispielsweise durch
    * Einbeziehung komplexerer Arten von Sudokus
    * Entwickeln eines Generators für Sudokus

Wenn es langweilig wird . . .

- kleine Happen für Zwischendurch:
  Tik-Tak-Toe, Vier-gewinnt, Mastermind, . . .
- Mensch gegen Mensch rasch ausprogrammiert
- Mensch gegen Computer auch einfach machbar
- für die ganz große Langeweile:
  Computer gegen Computer

Zeigen was man kann

- Problem: gute Programme können Freunde, Eltern, ...
  kaum überzeugen (die verstehen das nicht)
- Lösungsansatz: *Fraktale* berechnen und zeichnen
  (z.B. Ausschnitte aus Mandelbrot-Menge)
- Vorteil: rasch gemacht, wirkt immer gut
- Nachteil: man verfällt leicht dem Reiz der Bilder
  (statt die Schönheit guten Codes zu erkennen)

Schifahren & Co

- Problem: Simulation erzeugt nicht das richtige Feeling
- Ursache: „Um die reale Welt simulieren zu können muss man sie kennen.“
- Lösungsansatz: Computer ausgeschaltet lassen
- kurzfristiges Ziel: schöne Ferien!
- längerfristiges Ziel: mit Schwung in’s nächste Semester

Viel Glück beim abschließenden Test

Danke für Ihre Mitarbeit