Praktikum zu Informatik 2

Freudenreich, Paulus, Schröder

Einführung ins Informatik 2 Praktikum

In diesem Semester steht die Vertiefung der bisherigen C-Kenntnisse im Vordergrund. Neben der reinen Programmierung geht es um zwei weitere zentrale Aspekte der Softwareentwicklung:

- Arbeiten im Team: Zusammenarbeit über Versionsverwaltungssysteme (git).
- Einarbeitung in bestehenden Code: Weiterentwicklung und Anpassung von Projekten.

Über drei Praktikumseinheiten hinweg üben Sie den Umgang mit git, Makefiles und grundlegenden Testing-Konzepten. Der bereitgestellte Beispielcode implementiert ein Wortsalatspiel: Wörter werden aus einer Textdatei geladen und zufällig horizontal oder vertikal in ein Gitter gesetzt. Der Spieler muss diese Wörter finden.

Ihre Aufgabe ist es, das Projekt zum Laufen zu bringen, die fehlenden Bestandteile zu implementieren und sicherzustellen, dass die Lösung die bereitgestellten Tests besteht.

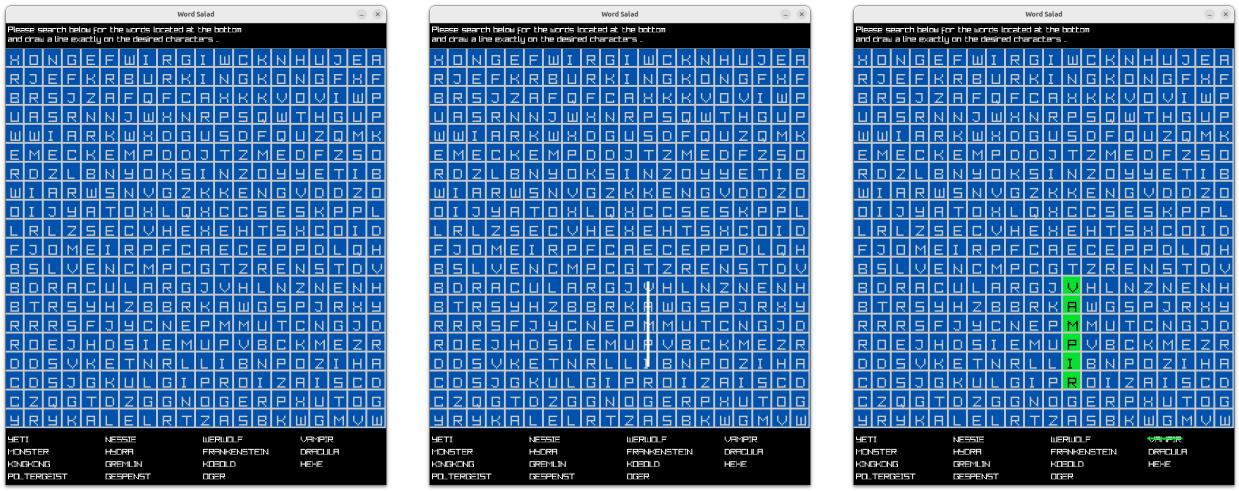
Einstieg in Git & Grundgerüst

Inhalte: Git-Basics, Branching, CI/Pipelines, Kompilieren mit Makefile, Modultechnik.

Aufgaben:

- Git vorbereiten
  - Lesen Sie die Unterlagen "EinführungInGit.pdf" und machen Sie sich mit git vertraut
  - Klonen Sie danach das Repository "Wortsalat" von efi git
  - Richten Sie sich ein eigenes Repo für Ihr Informatik2 Praktikum ein
- Projektstruktur erkunden
  - Welche Dateien sind Header, welche Implementierung?
  - Welche Funktionen sind bereits vorhanden, welche fehlen?
  - Schauen Sie die bereitsgestellten Screenshots an, um das erwartete Verhalten zu verstehen.
- Makefile & Initialversion
- Lesen Sie die Datei "Makefiles.pdf", kompilieren Sie das Projekt und testen Sie das (noch unvollständige) Spiel, indem Sie das Target `wordsalad_initial` bauen.
  - Hinweis: Hierbei wird das (mitgelieferte) Modul `libwordsalad_complete.a` verwendet. Dies enthält eine lauffähige Version des Spiels zum Testen (d.h. die `input.o`, `game.o` und `main.o` Objektdateien).

```
make wordsalad_initial // Bauen des Spiels, obwohl Dateien unvollständig
./wordsalad_initial words.txt  // Aufrufen des Spiels mit Datei
```



- Eigenes Target
  - Ergänzen Sie in main.c die Initialisierung des Spiels.
  - Erstellen Sie in Ihrem Makefile ein neues Makefile-Target, z. B. `wordsalad_myversion`, das Ihre eigene `main.c` enthält und zusammen mit der Bibliothek `libwordsalad.a` kompiliert (dies enthält die `input.o` und `game.o` Objectfiles).

Datenstrukturen & Spiellogik

Inhalte: Strings, char-Zeiger, Zeigerarrays, Enums, typedefs, structs, Pufferüberläufe

Aufgaben:

- Spiel-Logik in `game.c`
  - Platzieren Sie die Wörter aus der Wörterliste zufällig horizontal oder vertikal im Spielfeld.
  - Füllen Sie die übrigen Felder mit zufälligen Buchstaben.
  - Definieren Sie ggf. geeignete Datentypen (z.B. enum für Platzierungsrichtung etc.).
- Einlesen von Wörtern in `input.c`
  - Implementieren Sie eine Funktion, die ein Wort aus der Datei `words.txt` einliest und als C-String zurückgibt
- Achten Sie auf Speichersicherheit:
  - Nutzen Sie sichere Funktionen wie strncpy.
  - Übergeben Sie Arraygrößen immer als Parameter.

Tipp: Testen Sie Ihre Implementationen schrittweise und führen Sie nach jedem Teilabschnitt einen Commit durch.

```
make test && ./runTests  // unit tests bauen und laufen lassen
```

Feinschliff

- Stellen Sie sicher, dass das gesamte Projekt:
  - ohne Warnungen kompiliert,
  - alle Unit-Tests besteht

- Ergänzen Sie fehlende Kommentare und achten Sie auf einheitliche Formatierung (siehe Coding Guidelines).
- Optional: Erstellen Sie eine kurze README.md, die den Spielablauf und die Bedienung erklärt.