Portfolioprüfung – Werkstück A – Alternative 3

1 Aufgabe

Entwickeln Sie ein Schiffe-Versenken-Spiel mit Einzelspieler- und Mehrspielermodus mit „grafischer Darstellung“ in der Shell.

Für mehr Informationen zum Spiel und der Spielweise, siehe Wikipedia[1].

2 Anforderungen

- Benutzer sollen mit dem fertigen Spiel alleine gegen den PC und gegen einen menschlichen Mitspieler am gleichen PC spielen können.
- Das Spielfeld wird in der Shell angezeigt.
- Die Anzahl der Felder in der Breite und Höhe der zu generierenden Spielfelder gibt der Benutzer per Kommandozeilenargument an. Also z.B. `-xaxis 10 -yaxis 10`. Schränken Sie den Wertebereich sinnvoll ein.
- Die Anzahl der Schiffe pro Spieler und deren Größen und Namen können Sie im Programm fest einprogrammieren oder alternativ per Kommandozeilenargumente vom Benutzer definieren lassen.
- Wird gegen einen menschlichen Spieler am gleichen PC gespielt, wird abwechselnd das Spielfeld des jeweils an der Reihe befindlichen Spielers eingeblendet und der wartende Spieler muss solange wegschauen.
- Wird gegen den PC gespielt, erfolgen die Spielzüge des PCs via Zufallsfunktion.
- Mit der Tastatur (und evtl. auch mit der Maus) wählt der Benutzer zu Beginn des Spiels die Positionen seiner Schlachtschiffe aus.
- Der Spieler, der an der Reihe ist, definiert mit der Tastatur (und evtl. auch mit der Maus) auf welches Feld gefeuert werden soll. Alternativ soll es auch möglich sein, die Koordinaten einzugeben.
- Hat ein Spieler (oder der PC) alle Schiffe des Gegners versenkt, hat er das Spiel gewonnen. Dieses soll entsprechend angezeigt werden. Das kann beispielsweise durch eine Laufschrift geschehen, durch ein Blinken oder durch ein Invertieren der Farben in der Shell, etc.
- Der Spielstand, also die Anzahl der versenkten und noch übrigen Schiffe des aktiven Spielers und des Gegners, werden ständig über, unter oder neben dem

[1] `https://de.wikipedia.org/wiki/Schiffe_versenken`

Spielfeld des aktiven Spielers angezeigt. Es muss auch immer klar ersichtlich sein, welcher Spieler aktuell am Zug ist.

- Es soll jederzeit klar ersichtlich sein auf welche Felder man schon gefeuert hat und auf welche Felder der Gegner schon gefeuert hat.
- **Entwickeln und implementieren Sie Ihre Lösung als Bash-Skript, als C-Programm oder als Python-Skript** als freie Software (Open Source) und verwenden Sie hierfür ein Code-Repository, z.B. bei GitHub.
- Für die „grafische Darstellung“ in der Shell verwenden Sie eine Bibliothek wie **ncurses**[2] [3] (für C-Programme), **Termbox**[4] (für C-Programme oder Python-Scripte), **dialog**[5] [6] [7] (für Shell-Scripte) oder **Whiptail**[8] [9] [10] (für Shell-Scripte).
- Es soll möglich sein, das Spiel nur aus einer Shell heraus zu spielen. Es soll komplett in der Shell ablaufen!
- Spielen zwei menschliche Spieler an einem PC gegeneinander, kann es sinnvoll sein einen Countdown (z.B. 15 Sekunden) anzuzeigen, und wenn der menschliche Spieler in der Zeit kein Feld zum beschießen auswählt, feuert der PC via Zufallsfunktion.
- Spielt man gegen den PC, und hat dieser einen Treffer gelandet, soll dieser bei seinen nächsten Zügen versuchen bei umliegenden Feldern Treffer zu landen.
- Der PC als Gegner soll zu keiner Zeit ein Feld mehrmals beschießen.
- Eigene Boote dürfen sich beim Legen nicht überlappen. Gilt auch für den PC.
- Eventuell fallen Ihnen sinnvolle Erweiterungen ein. Um den Strategieaspekt zu erhöhen, wäre es z.B. sinnvoll, wenn jeder Spieler vor dem Feuern auf ein Feld angeben muss, mit welchem eigenen Schiff gefeuert wird und der nächste Spieler bekommt einen Hinweis aus welcher Himmelsrichtung gefeuert wurde.
- Der Quellcode soll durch Kommentare verständlich sein.
- Bearbeiten Sie die Aufgabe in Teams zu **3 Personen**.
- Schreiben Sie eine aussagekräftige und ansehnliche Dokumentation (Umfang: **8-10 Seiten**) über Ihre Lösung.

[2] http://openbook.rheinwerk-verlag.de/linux_unix_programmierung/Kap13-002.htm
[3] https://de.wikibooks.org/wiki/Ncurses:_Grundlegendes
[4] https://github.com/nsf/termbox
[5] http://openbook.rheinwerk-verlag.de/shell_programmierung/shell_007_007.htm
[6] https://www.linux-community.de/ausgaben/linuxuser/2014/03/mehr-komfort/
[7] https://linuxkurs.spline.de/Ressources/Folien/Linux-Kurs-7.pdf
[8] https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail
[9] https://saveriomiroddi.github.io/Shell-scripting-adventures-part-3/
[10] https://www.dev-insider.de/dialogboxen-mit-whiptail-erstellen-a-860990/

- Die Funktionalität der Lösung müssen Sie in der Übung demonstrieren. Bereiten Sie einen Vortrag mit Präsentationsfolien und eine Live-Demonstration (Umfang: **15-20 Minuten**) vor.