DOKUMENTATION

1. Festlegungen

Das Spiel kann von zwei Spielern oder von einem Spieler und einer KI gespielt werden.
Jeder Spieler hat ein Feld, auf dem er seine Schiffe setzt und auf das vom Gegenspieler geschossen wird. Diese Felder sind quadratisch, die Größe kann zwischen 10*10 bis 20*20 (Grenzen mittels symbolischer Konstanten einstellbar) gewählt werden.
Realisiert werden sie durch 2D-Matrizen vom Typ char. Die Werte in einer Matrix stehen für:
w = Wasser
M = Miss (auf diesen Punkt wurde geschossen, aber es befindet sich kein Schiff dort)
X = Hit (auf diesen Punkt wurde geschossen und es befand sich ein Schiff dort)
[zahl] = Schiff (der Länge [zahl] das an diesem Punkt noch nicht getroffen wurde)
D = Downed (Hier liegt ein Schiff, das komplett zerstört wurde)

Diese Matrizen werden zu internen Berechnungen verwendet. Bei der Ausgabe einer Matrix wird diese mit Hilfe von Ascii-Art verschönert und zensiert.

Die Anzahl und Länge der verfügbaren Schiffe wird vom Spieler festgelegt und in einem Array der Form: ships = [(Länge von Schiff 1), (Länge von Schiff 2), …] gespeichert.
Dabei muss die Länge eines Schiffes zwischen 2 und 7 liegen. (Grenzen mittels symbolischer Konstanten einstellbar)
Die Summe aller Schiffslängen darf einen gewissen Wert, der abhängig von der Spielfeldgröße ist, nicht überschreiten.

Der Spieler kann wählen ob er die Schiffe selbst setzen möchte, oder dies automatisch getan werden soll. Die KI setzt die Schiffe automatisch. Beim setzten der Schiffe gilt:

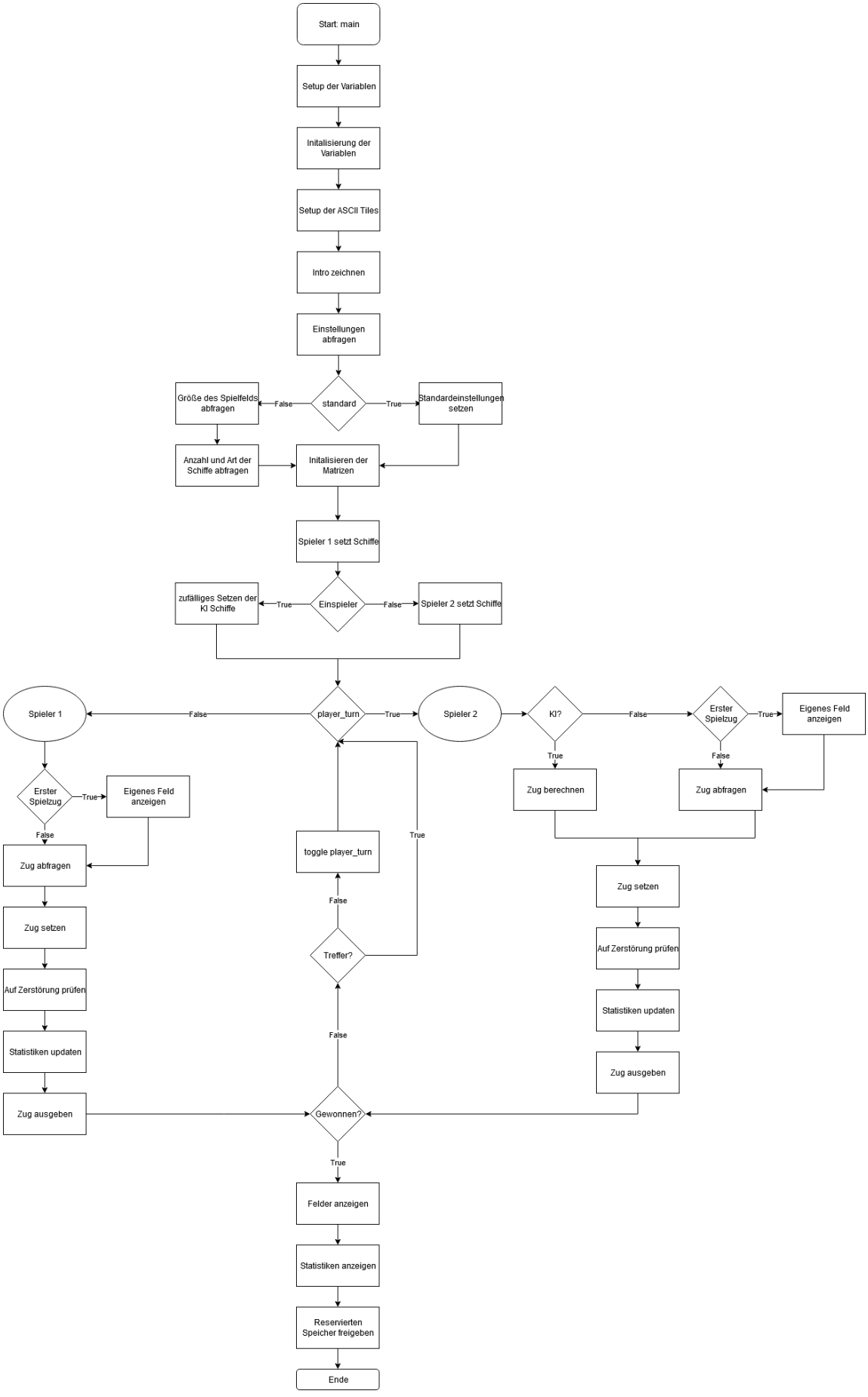
- Jedes Schiff muss genau einmal gesetzt werden.
- Ein Schiff wird entweder horizontal oder vertikal gesetzt.
- Ein Schiff muss komplett im Spielfeld liegen.
- Ein Schiff darf ein anderes nicht überschneiden.
- Ein Schiff darf ein anderes nicht berühren.

Beim Schießen auf einen Punkt gilt:

- Der Schuss muss im Spielfeld liegen.
- War der Schuss ein Treffer, darf der Spieler ein weiteres Mal schießen.

Im Programm werden zur Vereinfachung von Rechnungen Koordinaten in der Form y/x angegeben, in Ein- und Ausgaben in der Form x/y.

2. Programm-Ablauf-Plan



3. Die KI

Die Künstliche Intelligenz besteht aus 3 Schwierigkeitsstufen zwischen denen gewählt werden kann:

1. Leicht
2. Mittel
3. Schwer

Um den Spielzug der KI zu berechnen wird die Methode get_ai_turn() aufgerufen. Diese Leitet den Aufruf an die entsprechende KI weiter.

1. Leicht

Die KI schießt auf einen zufälligen Punkt im Spielfeld.

2. Mittel

Definition „sinnvoller Punkt“:

- Der Punkt liegt im Spielfeld.
- Auf diesen Punkt wurde noch nicht geschossen.
- Der Punkt liegt nicht neben einem versenkten Schiff.

Die KI sucht das Feld zunächst nach einem getroffenen aber noch nicht versenkten Schiff ab.
Es wurde kein getroffenes Schiff gefunden:

Die KI schießt auf einen zufälligen sinnvollen Punkt.

Es wurde ein getroffenes Schiff – Punkt gefunden:

Es kommen 4 Punkte in Frage.
Die KI schießt auf einen sinnvollen benachbarten Punkt.

Es wurden zwei oder mehr getroffenes Schiff – Punkte gefunden:

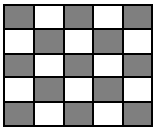
Diese Punkte müssen zusammenhängen, es kommen also 2 Punkte in Frage bei denen sich das Schiff fortsetzen kann.
Die KI schießt auf einen sinnvollen dieser zwei Punkte.

Auf diese Weise schießt die KI solange auf zufällige sinnvolle Punkte, bis sie ein Schiff gefunden hat. Sie arbeitet dann an diesem Schiff bis es versenkt wurde. Es gibt also auch immer nur ein getroffenes, aber nicht versenktes Schiff gleichzeitig.

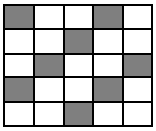
3. Schwer

Die KI verhält sich wie die zweite KI mit einer Erweiterung:
Beim Suchen nach einem neuen Schiff schießt sie nicht auf zufällige sinnvolle Punkte, sondern auf zufällige sinnvolle Punkte, die in einem Muster liegen, welches so gewählt ist, dass das kleinste Schiff im Spiel gerade nicht zwischen die Punkte in diesem Muster passt.

Beispiel kleinstes Schiff hat die Länge 2:

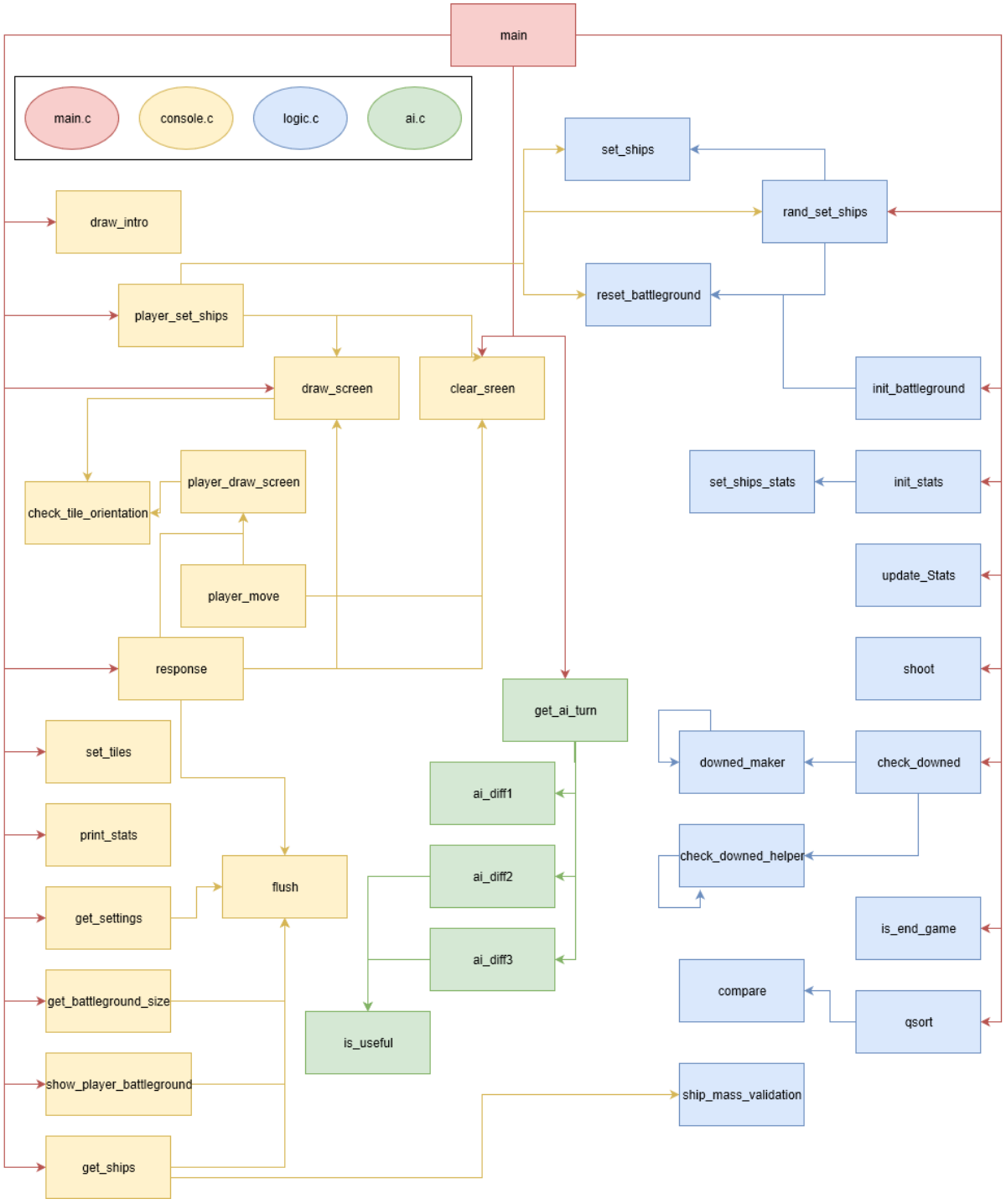


Beispiel kleinstes Schiff hat die Länge 3:



Auf diese Weise wird das gesamte Spielfeld mit der kleinstmöglichen Anzahl an Schüssen abgedeckt.

4. Programmstruktur



5. Beschreibung der Methoden in den verschiedenen Programmmodulen

ai.c:

- Int *get_ai_turn
  - Ruft je nach übergebener Schwierigkeit eine der drei Methoden zum Berechnen des Spielzugs auf
  - Gibt Array mit x, y Koordinaten aus
- Int *ai_diff1
  - Schießt auf zufällige Felder innerhalb von size
  - Gibt Array mit x, y Koordinaten aus
- Int *ai_diff2
  - Schießt auf ein sinnvolles Feld innerhalb von size, analysiert dazu die Matrix mithilfe von is_useful
  - Gibt Array mit x, y Koordinaten aus
  - Weitere Infos: Dokumentation Kapitel 3: Die KI
- Int *ai_diff3
  - Schießt auf ein sinnvolles Feld innerhalb von size, analysiert dazu die Matrix mithilfe von is_useful
  - Gibt Array mit x, y Koordinaten aus
  - Weitere Infos: Dokumentation Kapitel 3: Die KI
- Int is_useful
  - bestimmt abhängig von übergebener Schwierigkeit, ob ein Schuss sinnvoll wäre
  - gibt einen Wahrheitswert zurück

console.c:

- Void draw_intro
  - Gibt eine Begrüßung für den Spieler aus
- Void draw_sceen
  - Erstellt eine Matrix der Größe “size”, die vom Benutzer angepasst werden kann
  - Ruft “check_tile_orientation” auf (--> entscheidet so, ob Bug, Deck, oder Heck gesetzt wird)
- Void player_draw_screen
  - Zeichnet ein zweites, leeres Spielfeld, um die Schiffe des Gegners zu verstecken
  - Lediglich versenkte Schiffe, Treffer und Fehlschüsse werden angezeigt
- Int check_tile_orientation
  - Prüft die Umgebung des ausgewählten Feldes auf Wasser, oder einem schon beschossenen Feld
  - Übergibt das Ergebnis an “draw_screen”
- Void set_tiles
  - Initialisiert die graphische Darstellung der Schiffe
- Void print_stats
  - Gibt am Ende des Spiels die Statistik aus
- Void clear_screen
  - “Säubert” die Konsole durch das Ausgeben vieler \n Escape-Sequenzen
- Void player_set_ships
  - Lässt den Spieler zunächst entscheiden, ob die eigenen schiffe selbst setzten will:
  - Ja --> der Spieler wählt die Schiffart, dann die gewollte Koordinate und die Richtung, in der das Schiff gesetzt wird
  - Nein --> Eingabe wird an “rand_set_ships” übergeben
- Void get_settings
  - Spieler kann die Spieleinstellungen festlegen
  - Lässt zwischen Ein- und Zweispielermodus entscheiden
  - Ein Spieler: Schwierigkeitsgrad der K.I. kann aus drei Modi gewählt werden.
  - Zwei Spieler: Ausführen von “player_set_ship” für Spieler zwei
  - Spieler kann zwischen Standard- und Eigeneinstellungen wählen
- Int get_battleground_size
  - Es wird eine individuelle Feldgröße zwischen 10 und 20 festgelegt
  - Der Wert wird an “draw_screen” übergeben
- Int get_ships
  - Spieler wählt Anzahl der Schiffe (1 – 20)
  - Spieler wählt welche Art von Schiffen wie oft auftreten
- Int *player_move
  - Säubert die Konsole und gibt das Spielfeld aus
  - Spieler entscheidet, durch Angeben der Koordinaten, auf welches Feld er schießen möchte
- Void show_player_battleground
  - Zeigt dem Spieler das Eigene Feld mit den gesetzten Schiffen und durch den Gegner beschossene Felder
- Int response
  - Prüft ob ein Schiff getroffen wurde und gibt das Ergebnis aus
- Void flush
  - Puffer leeren

logic.c:

- Int init_battleground
  - Initialisiert eine Matrix der Größe size
  - Ruft reset_battleground auf
  - Gibt im Erfolgsfall 1, im Fehlerfall OUT_OF_MEMORY zurück
- Int init_stats
  - Initialisiert die Statistikspeicher je nachdem, welche Schiffe es gibt
  - Ruft hierzu auch set_ships_stats auf
  - Gibt im Erfolgsfall 1, im Fehlerfall OUT_OF_MEMORY zurück
- Void set_ships_stats
  - Wertet ships[] aus und extrahiert die Schiffstypen
- Void update_stats
  - Erhöht die Stats je nach getroffener Schiffsklasse
- Void reset_battleground
  - Setzt die Matrix auf den Anfangszustand (Wasser überall) zurück
- Int rand_set_ships
  - Setzt die Schiffe zufällig auf dem Spielfeld
  - Ruft hierzu set_ship auf
- Int set_ship
  - Überprüft die Position, an die das Schiff gesetzt werden soll und schreibt es in die Matrix, falls möglich
  - Gibt im Erfolgsfall 1, sonst 0 zurück
- Int ship_mass_validation
  - Prüft, ob die Anzahl an Schiffen auch in die vorliegende Matrix passt
  - Gibt im Erfolgsfall 1, sonst 0 zurück
- Int shoot
  - Schießt auf das Feld an den übergebenen Koordinaten
  - Gibt zurück welche Schiffsklasse getroffen wurde
- Int check_downed
  - Überprüft, ob ein Schiff versenkt wurde oder nicht und setzt dieses Schiff im Bedarfsfall auf versenkt
  - Ruft hierzu check_downed_helper (Hilfsmethode, hier nicht aufgeführt) und downed_maker auf
  - Gibt 1 oder 0 zurück
- Int downed_maker
  - Setzt ein Schiff rekursiv auf (D)owned
- Int is_end_game
  - Prüft ob das Spiel beendet ist (sich also noch Ziffern in der Matrix befinden)
  - Gibt 1 oder 0 zurück
- Int compare
  - Vergleichsfunktion für qsort in main() Vgl. http://www.cplusplus.com/reference/cstdlib/qsort/