Universität Heidelberg
Dr. Daniel Kondermann

15. Mai 2017
Programmierkurs

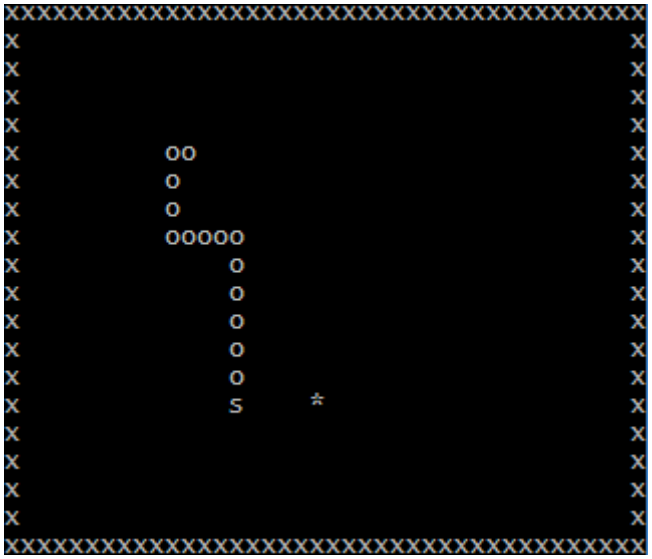
Aufgabenblatt 3

Abgabedatum: 22. Mai 2017

Jede Aufgabe dieses Blattes bringt 2 Punkte. Sie können insgesamt 10 Punkte erreichen. Packen Sie Ihre komplette Visual Studio Solution in **einem ZIP-Archiv**, das sie nach dem Muster '*SheetX-Surname.zip*' benennen und laden sie dieses im Moodle hoch

---

Das Ziel dieses Aufgabenblattes ist es ein einfaches, aber durchaus komplettes Snake-Spiel auf der Konsole zu implementieren. Snake ist ein Endlosspiel, bei dem man eine Schlange steuert und diese Futter fressen lassen muss. [1] Für jedes Stück Futter bekommt der Spieler Punkte und die Schlange wächst um ein Segment. Das Spiel ist beendet, wenn die Schlange entweder mit dem Rand kollidiert, oder mit sich selbst. Am Ende wird dies in etwa aussehen wie auf dem folgenden Screenshot:



Um diese Aufgabe etwas leichter zu gestalten, werden Sie nicht das komplette Spiel schreiben, sondern lediglich ausgewählte Teile - der Rest ist bereits fertig implementiert. Zu diesem Zweck laden sie sich bitte aus dem Moodle das Archiv '*Aufgabe3.zip*' herunter und entpacken es. Das Archiv enthält eine Visual Studio Solution und entsprechende Bibliotheken die benötigt werden. Wenn Sie die Solution öffnen, werden Sie wie üblich eine *main.cpp* finden, in der die Spielelogik implementiert ist. Die entsprechenden Stellen die bei den Aufgaben bearbeitet werden sollen sind jeweils mit einem Kommentar markiert. Zusätzlich werden Sie die Dateien *terminal.h* und *terminal.cpp* finden, welche einige Funktionen bereit stellen, um die Konsole zu manipulieren.

Lesen Sie sich ruhig erstmal alles gemütlich durch und finden heraus, wie alles funktioniert. Für alle Aufgaben sollte genügend Kontext vorhanden sein, der Ihnen zeigt, wie sie Funktionen etc. nutzen müssen, um zum gewünschten Ergebnis zu kommen.

[1] https://de.wikipedia.org/wiki/Snake

Aufgabe 1 Das Spielfeld

Wenn Sie das Projekt bauen und das erste mal ausführen, werden Sie mit einem etwas leeren Fenster begrüßt. Sie sehen nur den oberen Spielfeldrand und ein Sternchen an einer zufälligen Position. Das Sternchen stellt das Futter da, dass die Schlange fressen muss um immer weiter zu wachsen. Ihre erste Aufgabe ist daher in der Funktion `void draw_border()` den Rest des Spielfeldes, also den linken, rechten und unteren Rand zu ergänzen.

Aufgabe 2 Die Schlange

Nachdem wir nun das Spielfeld sehen, wäre es auch interessant unsere Schlange zu sehen. Ihre Aufgabe ist daher die Funktion `void draw_snake(std::vector<Vec2D>& snake)` zu implementieren. Jedes Segment der Schlange soll durch ein kleines ‘o‘ dargestellt werden, der Kopf durch ein kleines ‘s‘ (siehe Screenshot).

Aufgabe 3 Keyboard Input

Unsere Schlange können wir jetzt sehen und bestimmt haben Sie schon festgestellt, dass sie sich sogar schon bewegt. Ein Spiel bei dem man nichts tun kann ist allerdings langweilig. Drücken Sie einmal ‘w‘ im Spiel - sie werden feststellen, dass die Schlange sich nun nach oben bewegt. Jegliche andere Richtungswechsel fehlen allerdings noch. Implementieren Sie diese! Drückt der Spieler ‘a‘, soll sich die Schlange nach links bewegen, bei ‘d‘ nach rechts und bei ‘s‘ nach unten. Wenn sich die Schlange gerade nach links bewegt soll man die Richtung außerdem nicht direkt in die entgegengesetzte Richtung ändern können. Gleiches gilt für die anderen Bewegungsrichtungen.

Aufgabe 4 Kollisionen

Die Schlange kann bereits Futter aufsammeln und kann endlos wachsen. Allerdings kann die Schlange noch mit nichts anderem kollidieren, was das Spiel *etwas zu leicht* macht. Implementieren Sie daher die Funktion `bool check_collision(std::vector<Vec2D >& snake)` . Diese Funktion soll *true* zurück geben, falls die Schlange mit dem Rand des Spielfeldes oder mit sich selbst kollidiert. In jedem anderen Fall soll sie *false* zurück geben.

Aufgabe 5 Neustart

Wenn Sie es soweit geschafft haben, haben Sie ein funktionstüchtiges Snake. Glückwunsch! Allerdings müssen Sie jedes mal, wenn das Spiel vorbei ist das Programm neu starten, auch wenn der ‘Game Over‘-Screen was anderes behauptet. Das nervt! Implementieren Sie also jegliche Funktionalität die nötig ist, um dass Spiel neu zu starten, wenn der Spieler ‘e‘ drückt.