Professur für Algorithmen
und Datenstrukturen
Prof. Dr. Hannah Bast
Johannes Kalmbach

Programmieren in C++
SS 2022

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Übungsblatt 3

Abgabe bis Dienstag, den **17. Mai 2022** um **12:00 Uhr**

Bei diesem Übungsblatt geht es darum, eine vereinfachte Version des Spiels *Snake* zu implementieren. Dabei bewegt sich eine Schlange fester Länge mit einer konstanten Geschwindigkeit von 10 „Pixel“ (siehe unten) pro Sekunde über den Bildschirm und kann mithilfe der Pfeiltasten gesteuert werden. Das Spiel ist zu Ende, wenn die Schlange mit dem Bildschirmrand oder sich selber kollidiert oder wenn die *Escape* Taste gedrückt wird.

Aufgabe 1 (10 Punkte)

Schreiben Sie Dateien *Snake.h* und *Snake.cpp* mit den folgenden Funktionen. Für einige der Funktionen wurde bereits ganz oder teilweise Code in der Vorlesung geschrieben und über das SVN bereitgestellt. Die Aufgabe ist deswegen weniger Arbeit, als es auf den ersten Blick den Anschein haben mag.

Benutzen Sie für die folgenden Werte globale Variablen: die Dimensionen des Spielfelds, die aktuelle Richtung, die Positionen der einzelnen Glieder der Schlange (ein Feld für die y-Koordinaten und ein Feld für die x-Koordinaten, wie in der Vorlesung für die „Box“).

void initTerminal(): Initialisierung des Terminals und der Spielfelddimensionen mittels *ncurses*.

void initSnake(): Setzen der Anfangspositionen der Glieder der Schlange und der Richtung.

void drawPixel(int y, int x): Malen eines Pixels an die gegebene Position. Ein Pixel soll dabei aus einem Leerzeichen (optional zwei, siehe Aufgabe 3) mit geeigneter Hintergrundfarbe bestehen.

void drawBorder(): Malen des Spielfeldrands.

void drawSnake(bool mode): Malen bzw. Löschen der Schlange, je nach Wert des Arguments. Der „Kopf“ (erster Pixel in Bewegungsrichtung) der Schlange sollte eine andere Farbe haben als der Rest.

bool collidesWithBorder(): Berechnen, ob ein Glied der Schlange auf dem Rand liegt.

bool collidesWithSelf(): Berechnen, ob zwei Glieder der Schlange an derselben Position sind.

[Mit 10 Pixel pro Sekunde auf die nächste Seite vorschlängeln]

void moveSnake(): Bewegen der Schlange in die aktuelle Richtung um genau einen Pixel.

bool handleKey(int key): Verarbeiten des gegebenen Keycodes. Bei einer Pfeiltaste soll die Richtung entsprechend angepasst und *false* zurückgegeben werden (eine Bewegung in die Gegenrichtung, zum Beispiel nach links wenn die Schlange sich gerade nach rechts bewegt, sollte nicht möglich sein). Bei der Taste *ESC* soll *true* zurückgegeben werden.

Schreiben Sie für jede Funktion, die nichts auf den Bildschirm malt, einen Unit Test gemäß dem 6. Gebot. Diese Tests sollten in einer Datei *SnakeTest.cpp* stehen. Benutzen Sie für die Tests kein *ncurses*. Setzen Sie die Höhe und Breite des Bildschirms „von Hand“.

Aufgabe 2 (5 Punkte)

Schreiben Sie eine Datei *SnakeMain.cpp* mit einer *main* Funktion, die das Spiel auf die eingangs beschriebene Weise realisiert. Wenn das Spiel zu Ende ist, sollte der Bildschirminhalt nicht sofort verschwinden, sondern erst nach (erneutem) Drücken der *Escape* Taste. Der Code für die *main* Funktion sollte relativ kurz sein und vor allem die Funktionen benutzen, die Sie für Aufgabe 1 geschrieben haben.

Aufgabe 3 (0 Punkte)

Wenn Sie Spaß an der Aufgabe haben, können Sie das Programm auf vielerlei Arten erweitern. Sie können beispielsweise einen Pixel als zwei Leerzeichen malen, damit es quadratischer aussieht. Die Schlange kann mit der Zeit immer schneller werden oder immer länger oder beides. Oder zeigen Sie die Zeit an, die bereits vergangen ist. Oder spielen Sie den Trauermarsch von Chopin, wennn das Spiel zu Ende ist. Lassen Sie Ihrer Fantasie freien Lauf.

Aufgabe 4 (5 Punkte)

Schreiben Sie ein Makefile, das so weit wie möglich Patterns und automatische Variablen einsetzt, wie in der Vorlesung erklärt. Funktionen (wie *wildcard*, *basename*, etc.) können Sie einsetzen, müssen es aber nicht. Sie können dabei davon ausgehen, dass es einen Basisnamen <*name*> gibt, und die Funktionen in <*name*>*.cpp* stehen, das Hauptprogramm in <*name*>*Main.cpp* und die Tests in <*name*>*Test.cpp* . Es soll dabei wie schon für das Übungsblatt 2 unter allen Umständen immer nur so viel neu kompiliert / gelinkt werden wie nötig.

Laden Sie wie gehabt alle Code-Dateien und das Makefile in unser SVN hoch, in einem neuen Unterverzeichnis *blatt-03*. Es gelten weiterhin die 10 Gebote auf dem Wiki und bitte nehmen Sie die Ratschläge Ihres Tutors oder Ihrer Tutorin ernst. Laden Sie wie gehabt auch eine Datei *erfahrungen.txt* hoch (im Unterordner *blatt-03*), in der Sie kurz Ihre Erfahrungen mit dem Ü3 und der Vorlesung dazu beschreiben.