**8M**: Das 8-Minutes Spiel

Programmierdokumentation

1. Struktur

Die Struktur des Programmes sieht so aus:

**main**

**Funktionsbibliotheken**

console

control

**Spielbehandlung**

game\_launcher

game\_21

game\_catch

game\_guess

**Eintrittspunkt**

**Ausgangstritt**

**Systems Bibliotheken**

1. Illustration: Struktur des Programmes

Pfeilen symbolisieren die Funktion Rufen zwischen verschieden Header und Source File:

* **main**
* Bibliotheken
  + **console**
  + **control**
* Spiel
  + **game\_laucnher**
  + **game\_21**
  + **game\_catch**
  + **game\_guess**

1. main

Eintrittspunkt ist im main.c definiert, durch

int main(int argc, char \*argv[])

Diese main Funktion initialisiert zuerst den Spieler und dann das Bildschirm. Es benutzt die in console.h, control.h und game\_launcher.h Funktionen. Das „unendlich entwickelte“ Menu läuft bis der Benutzer in der gerufene menu() Funktion die Option Austreten wählt. Betriebslauf wird in game\_launcher diskutiert. Dieser Fall gibt das Programm 0 zurück und terminiert.

1. Bibliothek: console

Eine eigene Funktionsbibliothek die für Behandlung des Bildschirms sorgt. Diese Funktionen können die Ausgang färben, das Bildschirmfenster bewegen und Größe ändern und die gedrückten Tasten auswerten.

* 1. Konstanten

Konstanten von console sind in console.h definiert. Ihre Aufgabe ist die Grundzustände zu bestimmen.

* MAX\_ROWS, MAX\_COLS: bestimmen die maximale Breite und Höhe des Programmes (soll in Charakteranzahl gegeben werden)
* CHAR\_WIDTH, CHAR\_HEIGHT: definiert die in pixels gegebene Weite und Höhe eines Buchstabens
* DEBUG\_MODE: wenn dieses Wert 1 ist, schreiben Funktionen und das Software einige Debug Informationen an Bildschirm aus
* DEFAULT\_HEIGHT, DEFAULT\_WIDTH: wenn die Konsole Dimensionen einstellende Funktionen falsche Parameters bekommt, werden diese Werte benutzt. Auch bei der Initialisierung werden die Konstanten als Grundweite und Grundhöhe benutzt.
* Farben sind durch das enum COLORS definiert. Diese Konstanten sind nach den in Windows.h definierten Funktionen eingestellt
* die in KEYCODES definierten eigenen virtuellen Tasten werden in read\_key Funktion benutzt
  1. Funktionen

void clear\_screen();

löscht das Bildschirm

void redraw\_header(int points);

zeigt das Logo des Programmes an Bildschirm mit der als Parameter (points) gegebene Punkte

void init(int points);

initialisiert das Programm beim Start. Hier wird automatisch clear\_screen und set\_size mit der Grundwerten gerufen und wird den Charaktercode Tafel eingestellt

void set\_color(const int background, const int foreground);

nimmt die zwei Farben als vorher geschriebene COLORS integer Konstante und feststellt die Hinter- und Vorgrundfarbe der Konsole.

void set\_size\_and\_position(const int width, const int height,  
const int x, const int y);

void set\_size(const int width, const int height);

void set\_position(const int x, const int y);

ändert die Größe des Bildschirmfensters und bewegt in die als integer gegebenen Positionen (durch das HandleWindow Objekt der Konsole). 2 Aliasfunktionen sind definiert, damit die Zwei unabhängig verändert werden kann.

int get\_width();

int get\_height();

die zwei Hilf-funktionen kommen mit der aktuellen Weite und Höhe zurück, damit dynamisch Objekten platziert werden können.

void go\_to(const int x, const int y);

bewegt den Cursor in die (x;y) Punkt (mit der Hilfe von in windows.h definierten Handle Objekten), wo x die Spaltenzahl und y die Zeilenzahl ist.

int read\_key();

auf getch() basierende Funktion auswertet die gepresste Taste, und gibt zurück entweder die gedrückte ASCII Charakter oder die eigene KEYCODES Konstante wenn es kein gültige normale Taste ist (zum Beispiel Funktionstasten wie Return, Escape, usw.)

1. Bibliothek: control
   1. Globale Variabel

player\_name

wird in init\_player() initialisiert und speichert den Name der Spieler.

* 1. Funktionen

int menu();

zeigt das Menu an und kehrt mit der gewählten Möglichkeit zurück.

void init\_player();

biem Start wird init\_player gerufen, die nach den Name der Spieler fragt. Das Programm kontrolliert, ob in der letzen Position richtig \0 steht (sonst würde das Spiel später fehlen).

int read\_text\_file(char \*filename);

liest das als Parameter bekomme File (als Text) ein. Es soll relativ zu der läufende exe gegeben werden, weil die Funktion automatisch das aktuelle Verzeichnis hinzufügt!

char\* get\_cwd();

die get\_cwd Funktion liefert das aktuelle Verzeichnis mit der Hilfe von in der direct.h definierten getcwd.

1. Spielbehandlung
   1. Struktur

Alle eigenen Spiele sind in verschiedenen Files (mit c und h) definiert. Wenn main nach dem menu(), game\_launch ruft wird die Behandlung übergeht. Game\_launch sorgt für die Konfigurierung und den Start den Spielen. Sie darstellt auch die Regeln und die eventuell erscheinenden wichtigen Informationen. Nachdem die nötige Übergabevariabel aus stdin eingelesen wurden, wird die in Spielfile bestimmte play\_... (zum Beispiel: play\_21(int N, int \*points)) gerufen. Wenn das Spiel zu Ende ist, kommt das Programm in game\_launch zurück.

* 1. Jetzige Spiele

Spiele können relativ einfach später addiert werden, weil man nur weitere c und h Files entwickeln sollte. Aktuelle:

* game\_21.c, game\_21.h
* game\_catch.c, game\_catch.h
* game\_guess.c, game\_guess.h
  1. Bestenliste

Bestenliste ist durch highscore.c und highscore.h entwickelt. Vorherige Daten werden aus den Binärfiles im Verzeichnis highscore gelesen. Nach den Spielen werden die Spieler auf das Bildschirm geschrieben, die am meisten Punkten in der aktuellen Kategorie gesammelt haben. Das Ergebnis des aktuellen Spiels wird auch in den Files gerettet. Eine allgemeine Liste mit Spieler in allen Kategorien ist in der Hauptmenu zu erreichen.