Übung zum C/C++-Praktikum Fachgebiet Echtzeitsysteme

TECHNISCHE UNIVERSITÄT DARMSTADT

Übungen für den 6. Tag

Musterlösungs-/Microcontroller-Projekte in Eclipse importieren

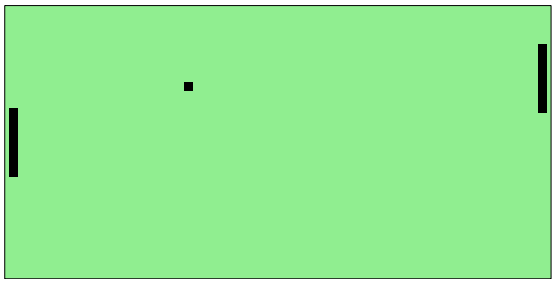
Die angebotenen Musterlösungs-/Microcontroller-Projekte basieren auf Makefiles. Daher ist es wichtig, dass du sie entsprechend importierst: **File -> Import...-> C/C++ / Existing Code as Makefile Project**.

Aufgabe 1 Implementation eines beliebigen Programms

Nachdem du alle Ein- und Ausgabemöglichkeiten des Boards kennengelernt hast, besteht die Aufgabe in diesem Teil darin, ein beliebiges Programm, zum Beispiel ein Spiel, zu implementieren. Du hast hierbei die freie Wahl, die folgenden Vorschläge sollen nur als Anregung dienen. Um die Aufgabe leichter zu gestalten, empfiehlt es sich die beiliegende `16FXlib` zu verwenden. Die Dokumentation dazu findest du unter `http://echtzeitsysteme.github.io/tud-cpp-16FXlib/documentation/index.html`.

Vorschlag: Pong

Zwei Gegner sollen je einen Balken (Rechteck) am linken oder rechten Rand des Spielfeldes mit den Schiebereglern steuern können, um einen Ball (ein Quadrat) im Spiel zu halten. Erreicht der Ball den linken oder rechten Rand des Spielfelds, so bekommt der Spieler auf der anderen Seite einen Punkt und der Ball wird an seine Anfangsposition (die Mitte des Spielfelds) zurückversetzt. Erreicht der Ball den oberen oder unteren Rand sowie einen der Balken der Spieler, so wird der Ball reflektiert - verlässt also niemals das Spielfeld.
Gewonnen hat der Spieler, der zuerst eine definierte Anzahl an Punkten erreicht. Die aktuelle Punktzahl beider Spieler könnte auf der Siebensegmentanzeige ausgegeben werden.



Vorschlag: Game of Life

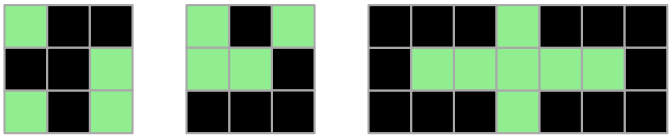
„Game of Life“[1] besteht aus einem zweidimensionalen Spielfeld. Jedes Feld steht für eine Zelle, die *tot* (grün) oder *lebendig* (schwarz) ist. Jede Zelle hat acht Nachbarzellen, die ebenso tot oder lebendig sein können. Zu Beginn gibt es eine vordefinierte Anfangsgeneration.
Durch festgelegte Regeln wird die nachfolgende Generation ermittelt:

- Eine **lebende Zelle** ...
  - mit 1 oder 0 lebenden Nachbarn stirbt aus Einsamkeit.
  - mit 4 oder mehr lebenden Nachbarn stirbt wegen Übervölkerung.
  - mit 2 oder 3 lebenden Nachbarn bleibt am Leben.

[1] siehe auch `http://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens`

- Eine **tote Zelle** mit genau 3 lebenden Nachbarn wird in der nächsten Generation geboren werden, andernfalls bleibt sie tot.

Als Anfangsgeneration eignen sich zufällige Populationen oder eine der folgenden Figuren:



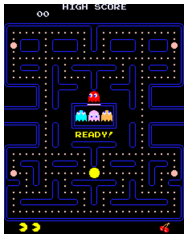
Hinweise

- Da das Spielfeld begrenzt ist, soll es torusförmig aufgebaut werden. Das heißt: Alles, was am unteren Rand des Spielfelds verschwindet, kommt oben wieder heraus – das Gleiche gilt für den linken und rechten Rand.
- Verwende als Spielfeld ein mehrdimensionales Array
- Ein weiteres mehrdimensionales Array bietet sich an, um die zukünftige Generation erzeugen zu können.
- Achte beim torusförmigen Feld unbedingt darauf, dass du nicht über die Grenzen des Spielfelds hinaus zugreifst! Das kann zu unvorhersehbarem und schwer zu debuggenen Verhalten des ganzen Displays führen!

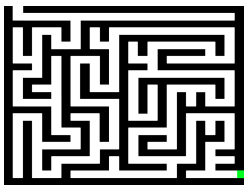
Weitere Vorschläge

Asteroids

Pacman



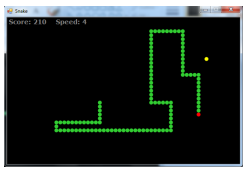
Labyrinth



Ausweichspiele à la Hugo

Moorhuhn

Snake



Hinweise

- Möchte man ein Bild auf dem Board anzeigen, so empfiehlt es sich die einzelnen Pixeldaten in einem Array zu speichern. GIMP bietet hierfür die Möglichkeit eine Bilddatei als Header im `GIMP header image file format` zu exportieren. Da das Board nur zwei Farbwerte ermöglicht, sollte das Bild vor dem Exportieren über den Menüpunkt **Bild –> Modus** auf **indiziert ...** (Schwarz/Weiß-Palette (1-Bit)) gestellt werden. Anschließend kann man das `header_data` Array verwenden, um auf die Bilddaten zuzugreifen.