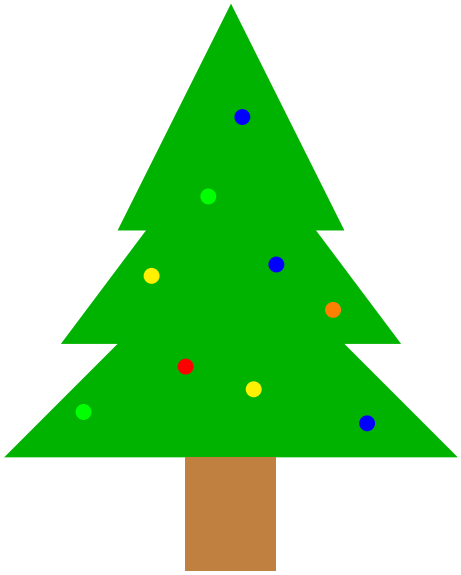
Weihnachtsaufgaben

Version 1 — 20. Dezember 2024

Name Last



Frohe Weihnachten und einen guten Rutsch ins neue Jahr!

Inhaltsverzeichnis

Motivation

Programmieren zu lernen ist ein Prozess, der vor allem durch Ausprobieren und Anwenden geschieht – "learning by doing". Genau das ist der Grund, warum ich euch diese Aufgaben über die Ferien mitgebe. Durch kleine Projekte könnt ihr eure Programmierfähigkeiten weiterentwickeln und könnt mit euren Ergebnissen herumspielen. Ich hoffe, dass ihr Spaß an den Aufgaben habt, und dabei auch etwas lernt!

1 Der perfekte Weihnachtsbaum

Die kalte Winterzeit hat das Dorf erreicht und die ersten Schneeflocken haben sich leise auf den Dächern niedergelassen. Inmitten dieser festlichen Atmosphäre kommt eine Gruppe von Dorfbewohnern auf die Idee, den schönsten Weihnachtsbaum zu erschaffen – allerdings nicht irgendeinen Baum, sondern einen digitalen Baum, der auf dem Bildschirm erstrahlen soll.

Die Dorfbewohner sind überzeugt, dass ein solcher Baum den Dorfplatz nicht nur verschönern, sondern auch die Herzen aller beflügeln wird. Sie haben von deiner Fähigkeit gehört, mit Code Wunder zu vollbringen, und so wenden sie sich an dich mit ihrem besonderen Wunsch: Sie möchten einen Weihnachtsbaum, der nicht nur in ASCII-Grafik erstrahlt, sondern auch die Magie der Festtage spüren lässt.

Mit einer klaren Vorstellung und einer Menge Ideen wollen sie, dass du diesen Baum erschaffst. Doch je mehr du dich mit dem Baum beschäftigst, desto mehr Wünsche erreichen dich. Es kommen Vorschläge, wie der Baum noch schöner, festlicher und kreativer gestaltet werden kann. Am Ende soll er nicht nur ein gewöhnlicher Baum sein, sondern ein wahres Meisterwerk der digitalen Kunst!

1.1 Aufgabe

Dein Programm soll einen Weihnachtsbaum zeichnen, dessen Höhe vom Nutzer bestimmt werden kann. Der Baum besteht aus einer symmetrischen Spitze und mehreren Ebenen, die nach unten hin breiter werden. Am Fuß des Baums befindet sich ein stabiler Stamm.

```
  *
 ***
*****
  *
```

Weihnachtsbaum mit Höhe 3

1.2 Erweiterungen

Die Dorfbewohner sind schon begeistert von dieser Grundversion, aber sie haben weitere Vorschläge gemacht, um den Baum noch beeindruckender zu gestalten.

- Dekorationen

  Der digitale Weihnachtsbaum soll nicht nur aus seinen grünen Zweigen bestehen, sondern auch festlich geschmückt werden, um die Dorfbewohner wirklich in Weihnachtsstimmung zu versetzen. Die Dorfgemeinschaft hat bereits einige Vorschläge gemacht, um den Baum noch prächtiger zu gestalten. Dein Auftrag ist es, diese Dekorationen hinzuzufügen, sodass die Dorfbewohner den Baum individuell gestalten können.

  - Schnee (~)

    Die kalte Winterluft hat sich im Dorf ausgebreitet, und nun beginnt der erste Schnee zu fallen. Feine Flocken wirbeln vom Himmel und setzen sich sanft auf den Baum. Lasse den Schnee durch die Luft tanzen und den Baum in eine winterliche Kulisse verwandeln.

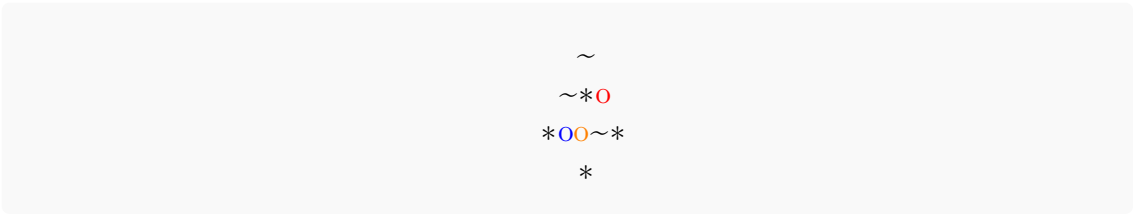
(→)

– Geschenke (`.`)

  Die Dorfbewohner haben liebevoll Päckchen vorbereitet und unter den Baum gelegt. Sorge dafür, dass sich einige Geschenke darunter befinden, die ein Hauch von Magie und Überraschung versprechen. Die Kinder werden sich freuen, wenn sie beim Blick auf den Baum sehen, dass ihre Geschenke bereits darauf warten, ausgepackt zu werden.

– *Lichterkette (`o`)*

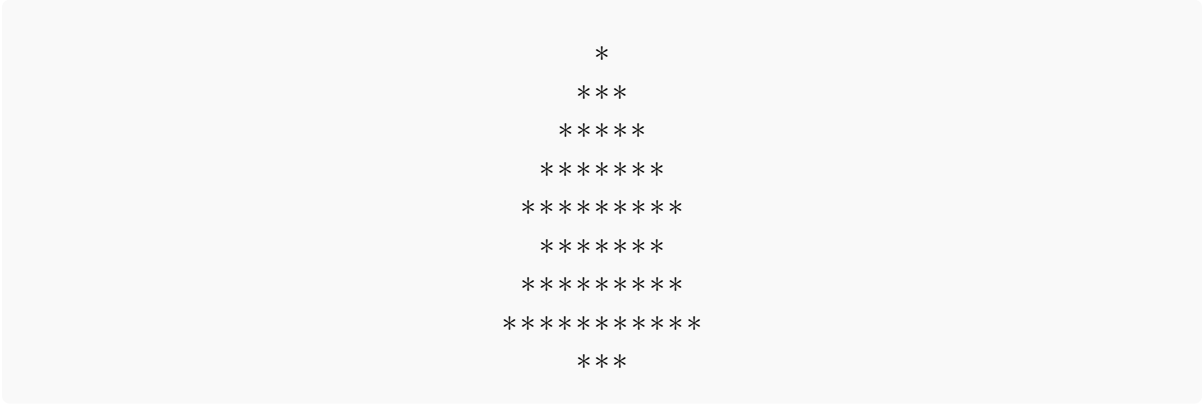
  Es wird Zeit, den Baum zum Leuchten zu bringen! Die Dorfbewohner haben eine Lichterkette aufgehängt, die in bunten Farben erstrahlt. Doch die Lichter sollen nicht nur leuchten, sondern auch tanzen! Lasse die Lichterkette in einer sanften Animation flimmern, sodass der Baum immer wieder in neuem Glanz erstrahlt. Der glitzernde Baum wird die Dorfbewohner in Staunen versetzen und den Dorfplatz in ein magisches Licht tauchen.



Weihnachtsbaum mit Schnee und Lichterkette

- Andere Baumarten

  Die Dorfbewohner haben begonnen, ihre eigenen Vorstellungen vom perfekten Baum zu äußern. Einige wünschen sich einen Baum, der sich von der klassischen Form unterscheidet. Diese Baumarten sollen auffälliger und markanter wirken, mit schärferen, spitzeren Zweigen. Lasse verschiedene Baumdesigns entstehen, damit jeder Dorfbewohnerin ihren oder seinen eigenen Baum erschaffen kann.



Ein anderer Weihnachtsbaum

- Zufällige Bäume

  Einige Dorfbewohner lieben es, sich überraschen zu lassen. Sie möchten einen Baum, der nicht immer gleich aussieht, sondern jedes Mal anders. Lasse daher zufällig generierte Bäume entstehen, die für jede Eingabe einzigartig sind.

2 Conway's Game of Life

Im Jahr 1970 entwickelte der britische Mathematiker John Conway das „Spiel des Lebens“ – ein zelluläres Automatenmodell, bei dem die Entwicklung von Zellen ausschließlich durch eine einfache Reihe von Regeln bestimmt wird. Jede Zelle kann entweder lebendig oder tot sein und verändert sich im Laufe der Zeit basierend auf den Zellen in ihrer direkten Nachbarschaft. Es handelt sich dabei um eine Simulation ohne äußere Eingriffe, in der das Verhalten der Zellen ausschließlich durch ihre Umgebung beeinflusst wird.
Das „Spiel des Lebens“ erregte rasch das Interesse von Wissenschaftlern und Mathematikern, da es auf faszinierende Weise komplexe Muster aus einfachen Regeln entstehen lässt. Zahlreiche Forscher begannen daraufhin, das Modell zu simulieren, um zu untersuchen, wie sich aus diesen simplen Regeln überraschende und teils lebendig wirkende Strukturen entwickeln können.

2.1 Aufgabe

Du wirst eine Simulation von Conway's Game of Life implementieren. In dieser Simulation gibt es ein zweidimensionales Gitter, in dem Zellen nach den oben beschriebenen Regeln leben, sterben oder sich fortpflanzen. Deine Aufgabe besteht darin, das Verhalten der Zellen über mehrere Zyklen hinweg zu simulieren, basierend auf den folgenden Regeln:

- Eine lebendige Zelle bleibt am Leben, wenn sie genau zwei oder drei lebendige Nachbarn hat. Andernfalls stirbt sie (entweder durch Überbevölkerung oder durch Einsamkeit).
- Eine tote Zelle wird wiederbelebt, wenn sie genau drei lebendige Nachbarn hat.

Die Simulation läuft über mehrere Zyklen hinweg und zeigt, wie sich das Muster der Zellen mit der Zeit entwickelt.

2.2 Erweiterungen

Nachdem du die Grundversion des Spiels implementiert hast, gibt es einige spannende Erweiterungen, die das Spiel noch interaktiver und interessanter machen können:

- Anpassbare Feldgröße

  Der Benutzer soll zu Beginn der Simulation die Größe des Spielfelds festlegen können.

- Geschwindigkeit anpassen

  Der Benutzer soll die Geschwindigkeit der Simulation anpassen können.

- Simulation pausieren

  Der Benutzer soll die Simulation jederzeit pausieren können.

- Zellen platzieren

  Der Benutzer soll Zellen manuell auf dem Spielfeld platzieren können, entweder durch Eingabe von Koordinaten oder mithilfe eines grafischen Interfaces.

- Zufällige Startmuster

  Der Benutzer soll die Simulation mit einem zufälligen Startmuster beginnen können.

I Formatierung mit ANSI

In der Welt der Terminal-Programmierung ist ANSI eine weit verbreitete Möglichkeit, Text mithilfe von Steuersequenzen zu formatieren. Mit ANSI-Codes können Textfarben, Hintergrundfarben, Formatierungen wie Fett oder Kursiv und vieles mehr festgelegt werden. Diese Codes sind besonders nützlich, um die Benutzererfahrung zu verbessern, indem sie Text visuell hervorheben.
Ein Beispiel für eine ANSI-Farbsequenz ist: `\033[31mDieser Text ist rot\033[0m.`
In diesem Beispiel bewirkt `\033[31m`, dass der Text rot wird, und `\033[0m` setzt die Formatierung zurück.
Für eine vollständige Liste der verfügbaren ANSI-Codes, inklusive Farben und Formatierungen, kannst du die folgende Ressource einsehen:
`https://gist.github.com/fnky/458719343aabd01cfb17a3a4f7296797`

II Formatierung mit den Prog1Tools

Ihr könnt davon ausgehen, dass die Methoden alle mit folgendem Beispiel arbeiten:

```
TextScreen screen = TextScreen.getInstance();
screen.setTitle("My cool documentation");

// Schreibt den Text "Hallo" an die Position
screen.write(y, x, "Hello");
```

Text schreiben

```
screen.write(int y, int x, String text);
```

Text einfärben

```
screen.setForegroundColor(int y, int x, Color color);
screen.setBackgroundColor(int y, int x, Color color);
```

Ihr könnt verschiedene vorgefertigte Farben benutzen:

```
Color.white
Color.lightGray
Color.gray
Color.darkGray
Color.black
Color.red
Color.pink
Color.orange
Color.yellow
Color.green
Color.magenta
Color.cyan
Color.blue
```

Ihr könnt außerdem RGB-Farben benutzen:

```
new Color(int r, int g, int b);
```

Bildschirm löschen

```
screen.clearScreen();
```


(→)

Verzögerung erstellen

```
TextScreen.pause(int timeInMs);
```

Gedrückte Taste abfragen
Das zuletzt gedrückte Zeichen bekommt ihr durch:

```
char c = screen.getSelectedKeyChar();
```

Falls ihr diesen Code in einer Schleife ausführt, so wird sich das Zeichen erst ändern, wenn der Benutzer ein weiteres drückt.
Um dem vorzubeugen, könnt ihr das Zeichen nach dem Lesen wieder löschen durch:

```
screen.resetKeyEvent();
```

Beachtet bitte, dass "kein Zeichenäls Leerzeichen dargestellt wird. Somit kann dies nicht genutzt werden, um die Leertaste abzufragen.
Falls ihr trotzdem das Leerzeichen abfragen wollt, so könnt ihr folgenden Code benutzen:

```
Character c = screen.getSelectedKeyChar();
if (c == '␣' && screen.getSelectedKeyCode() == 0)
    c = null;

if (c == null); // kein Zeichen
if (c != null); // Leerzeichen
```

Mausposition abfragen
Die Mausposition ist nur für das ganze Fenster selbst bestimmbar. Sprich, es ist nicht direkt möglich, das geklickte Zeichen abzufragen.
Jedoch versucht folgender Code, die richtige X/Y- bzw. Reihen-/Zeilenposition zu erkennen:

```
var pt = screen.getMousePosition();
if (pt != null) {
    int pxlX = (int) ((double) pt.x / (double) screen.getWidth() * 80 +
    int pxlY = (int) ((double) pt.y / (double) screen.getHeight() * 25
```

Ihr müsst hier darauf achten, ob ‘pt == null‘ ist. Falls dem so ist, ist die Maus nicht auf dem Fenster.
Die Variablen ‘pxlX‘ und ‘pxlY‘ sind hierbei die Position des Zeichens, über welches der Benutzer gerade hovert.

Damit habt ihr nun die grundlegenden Funktionen zur Verwendung des ‘TextScreen‘ in der Prog1Tools.jar kennengelernt. Viel Spaß beim Programmieren! - *Noah*

1 Contributors

Danke Noah für die Dokumentation zu den Prog1Tools!
(Ich schreib das noch besser, versprochen!)