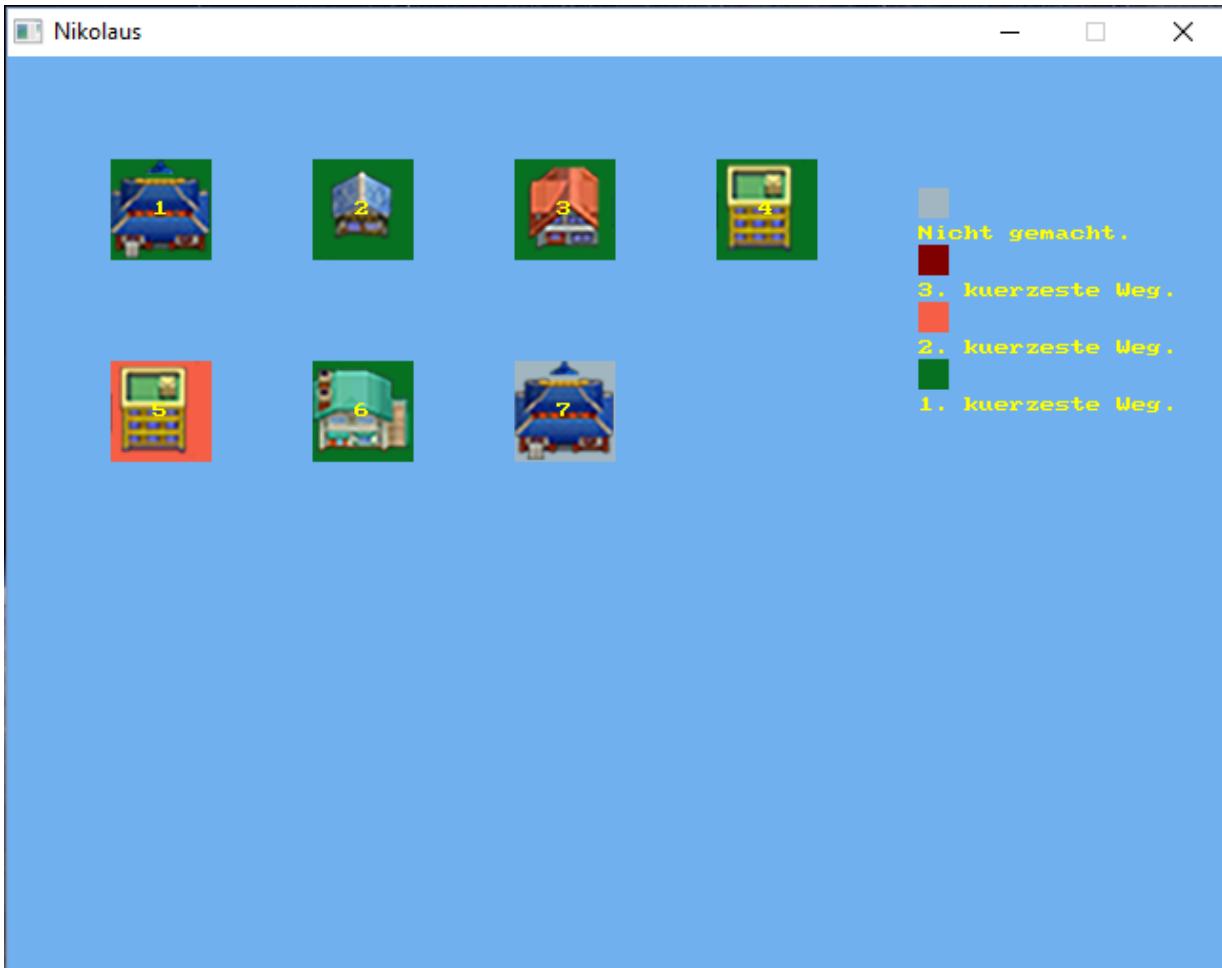


Benutzerdokumentation

I habe ein logisches Spiel verwirklicht. Es basiert auf ein Grundproblem, das Problem des Handlungsreisenden. Die Aufgabe ist der kürzeste Hamilton-Kreis in einem gerichteten Graph zu finden. Aber es ist sehr schwer zu finden, es gibt keinen Algorithmus, das es in polynomiale Zeit durchführt. Das Spiel hat ein graphisches UI, das ist mit SDL(Simple Directmedia Layer) gelöst. Das UI steht aus 2 Zustand: Menu-Zustand und Spiel-Zustand.

Das Menu sieht so aus:



Also an dem linken Seit stehen die Levels und an dem rechten Seit zeigen sich die Hilfsinformationen. Was aussagt welche Farbe was bedeutet.

grün: Beste

hellrot: 2. Beste

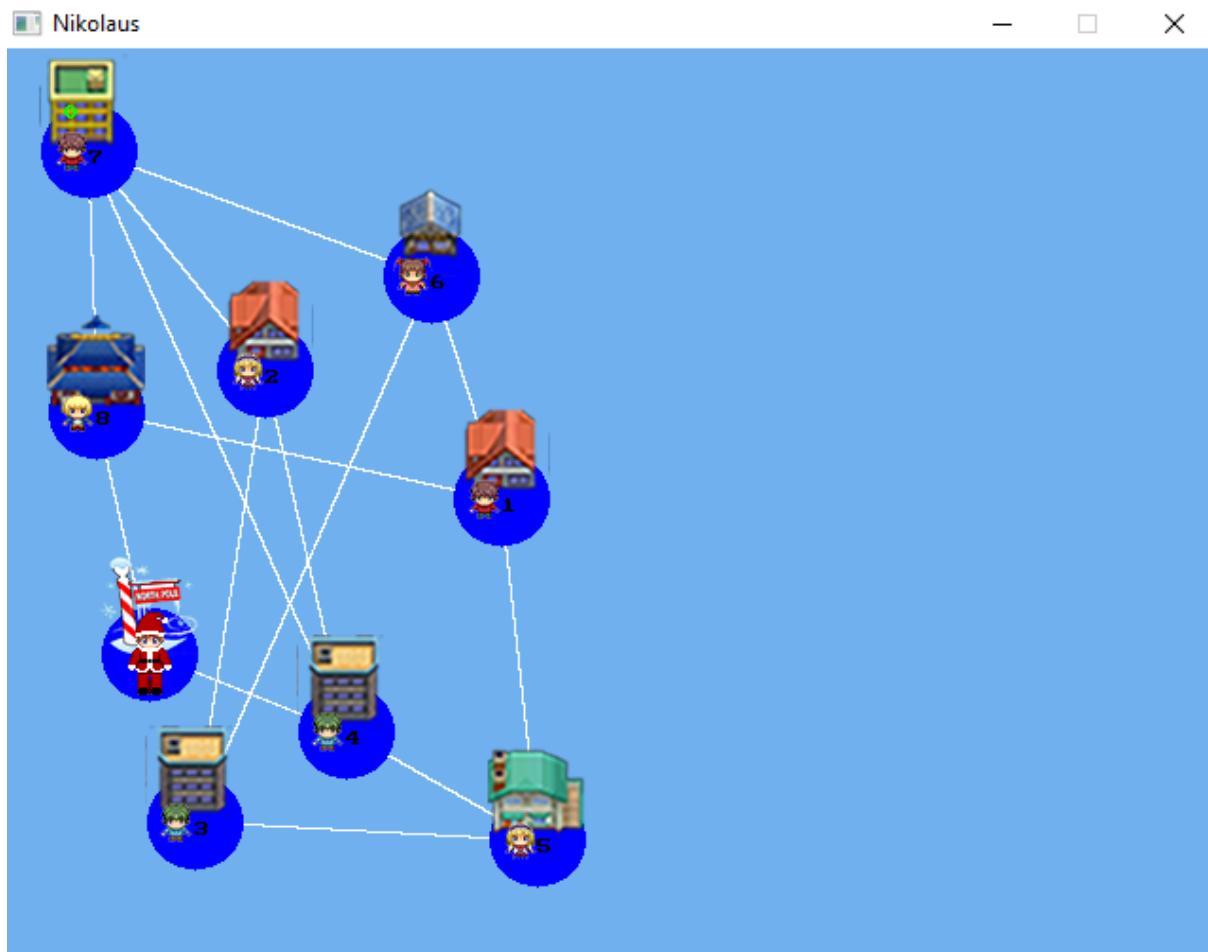
rot: 3. Beste

grau: Diese Level ist noch nicht gemacht haben

Es gibt 16 verschiedene Mappe, man kann nur dann die nächste ausprobieren, falls die letzte Level mindestens 'rot' gemacht wurde. Die Schwierigkeit inkrementiert als wir weitergehen. Also die erste ist ziemlich einfach, aber die letzte ist komplizierter, aber noch lösbar als Mann.

Man kann das ganze Spiel nur mit Maus steuern. Alles wirkt mit linken Klicken, deshalb braucht man keine Tastatur für dieses Spiel zu spielen. Wenn man einen Level auswählt, dann ladet das Programme die ausgewählte Level und wechselt der aktuelle Zustand zu Spiel-Zustand.

Das Spiel-Interface sieht so aus:



Nikolaus beginnt auf dem Nordpole. Das Ziel des Spielers ist mit Nikolaus jede Kind zu besuchen und dann wiederkehren zu Nordpole. Das Gewicht der Kanten ist die Entfernung von der Häuser. Der Weg ist die Summe des Gewichts. Ziel ist der kürzeste Weg zu finden.

Aber es gibt ein paare Regeln:

- Nikolaus kann nur die Nachbaren von einem Haus fahren (also existiert ein gemeinsame Kante)
- Nikolaus kann jede Kind nur einmal besuchen (also er kann nicht einen Knoten zweimal besuchen)
- Nikolaus muss am Ende auf dem Nordpole stehen

Man verliert, falls

- Nikolaus wird beschränkt sein (nicht auf dem Nordpole)
- Nikolaus geht mehr Weg als die 3. kürzeste Weg
- Nikolaus geht wieder zu Nordpole, aber es gibt Kind, das keine Geschenke bekommen habe

Nachdem der Spieler den Level beendet hat, können zwei Ausgänge sein:

- a. Man verliert, weil man zu lang Weg gegangen hat
- b. Man gewinnt, weil sein Weg ist niedriger als die 3. Weg (1/2/3)

Wenn man verliert oder den Level beendet hat, springen das Programm zu Menu-Zustand zurück mit einer schriftlichen Information, was gewesen ist. Falls man verliert, wird das score.txt nicht

verändern, aber falls man den Level gelöst hat, dann falls seine Ergebnis ist besser als was im Text-File ist, dann wird die ältere Ergebnis mit den neuen, besseren Ergebnis übergeschrieben.

Über den Graphics

Es gibt 4 verschiedene Bilder, die in diesem Spiel benutzt werden. Erste ist der Nikolaus von 4 verschiedenen Ansichten (Voransicht, Rückansicht, link Seitenansicht, recht Seitenansicht). Für jeden Ansicht gehören 4 verschiedene Bilder, die benutzt sind, den Gehen-Animation zu verwirklichen. Die Teilbild haben 32x48 Resolution. Zweite ist die Kinder (mit 8 Teilbilder). Diese Teilbilder haben 24x24 Resolution. Das Dritte Bild ist die Häuser. Dieses Bild beinhaltet 10 Teilbilder von 10 verschiedenen Häusern. Resolutionen dieser Häuser sind 52x52. Das letzte Bild ist der Nordpol selbst mit 52x52 Resolution. Alle Bilder sind in PNG-Datentyp gespeichert und einliest in dem Programm durch SDL-Funktionen.

Wenn man das Spiel ermüdet, dann kann man das Spiel mit den 'X' logo schließen.

Über das Hilfsprogramm

Das Hilfsprogramm verwirklicht einen brute-force Algorithmus, mit zurücktretende Suchen. Damit werden alle Hamilton-Kreis gefunden. Während des Suchens wird auch ein Minimum Suchen durchführen. Man kann dieses Hilfsprogramm von Terminal benutzen.

Man soll 2 Argumenten eingeben:

1. Eingang: Roh Graph Filename zB.: ,test.txt'
2. Ausgang: Fertig spielbar Karte Filename zB.: ,test_out.txt'

Wenn man nicht 2 Argumente eingibt, dann werdet das Programm Fehler ausschreiben. Diese werden so aussehen wie in der Spezifikation geschrieben hat.

Also:

N	M
x_1	y_1
x_2	y_2
...	...
x_i	y_i
von_1	zu_1
von_2	zu_2
...	...
von_j	zu_j

Das Ausgang-File sieht fast gleich aus. Aber am Ende werden die 3 kürzesten Hamilton-Kreiswege ausgeschrieben.