Aufgabe 3: Zauberschule

Team-ID: 00798

Bearbeiter dieser Aufgabe: Heinrich Fahrenbruch

17. November 2022

Inhalt

Lösungsidee

Die Fläche des Labyrinths soll erstmal aus der Text Datei ausgelesen und der Grundriss in Zwei Stockwerke aufgeteilt werden. Man benötigt ebenfalls die Informationen wo sich der Startpunkt und wo sich der Zielpunkt befindet und diese muss man ebenfalls in einer Variable speichern.

Wenn man alle Informationen zusammengetragen hat, wird das Labyrinth durch einen Flood Fill Algorithmus vom Ziel aus geflutet und jedes Feld bekommt einen Wert zugeteilt. Beide Stockwerke sollen separat geflutet werden.

Zuletzt wird dann ein Algorithmus eingesetzt, der den schnellsten Weg vom Startpunkt „A“ Zum Zielpunkt „B“ sucht, indem er sich an der kleinsten verfügbaren zahl in seinem Umkreis entlanghangelt, bis er am Ziel ist. An den Stellen, an denen sich die Zahlen vom oberen und vom unteren Stockwerk überschneiden, wird geprüft, ob der Wechsel sinnvoll ist. Für jedes Feld, welches er geht, soll er sagen in welche Richtung es gehen soll.
Am Ende wir dann der richtige Lösungsweg in der Konsole ausgegeben.

Umsetzung

Die Lösungsidee wird in C# implementiert. Weil C# eine Objektorientierte Programmiersprache ist, kann man gut die einzelnen Koordinaten als Objekte erstellen und behandeln, weil sie dadurch nachvollziehbarer dargestellt sind.

Softwarearchitektur

Um mir das gesamte Projekt übersichtlicher aufzubauen habe ich zuerst auf der Internetseite „play.d2lang.com“ ein Diagramm erstellt, bei dem man die Zusammenhänge zwischen den einzelnen Klassen und Methoden erkennen kann. Das anschließend fertige Diagramm habe ich in dem Ordner „D2“ im Projektordner abgelegt. Dort befindet sich auch der Code, welchen man im Live-Editor der Website einfügen kann und das Diagramm wird ebenfalls nochmal ausgegeben.

Auch wenn die gesamte Projektstruktur sich am Ende sehr vom anfänglichen Ansatz unterscheidet, ist es dennoch sehr hilfreich dabei gewesen, ein Fundament für das Projekt aufzubauen und einen guten Start zu schaffen.

Auslesen der Zauberschule

Am Anfang soll ein Objekt erstellt werden, namens Schule, welches die zauberschule.txt in die beiden Stockwerke aufteilt. Die schule beinhaltet eine Klasse namens „InitialisiereSchule“, welche den ganzen Code für die Initialisierung auslagern soll, um die Klasse „Schule“ übersichtlicher zu machen.

Zudem werden Zwei Stockwerke aus der Klasse „Stockwerk“ erstellt welche die Länge und Breite des Labyrinths so wie den Grundriss des jeweiligen Stockwerks als String Array enthalten. Ebenfalls habe ich zwei Booleans hinzugefügt welche überprüfen sollen, ob sich in jenem Stockwerk das Ziel und die Position der Person (der Start) befindet.

Es wird auch eine Person und eine Ziel Klasse erstellt, welche die Ursprünglichen Koordinaten der Person und des Ziels Speichern sollen. Die Koordinaten werden als PositionX und PositionY gespeichert.

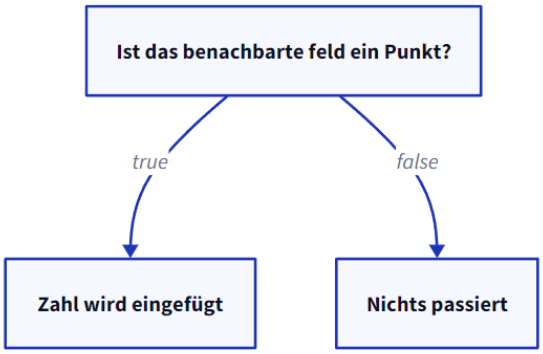
Flood Fill Algorithmus

Im Flood Fill Algorithmus werden die umliegenden Koordinaten vom Ziel ausgehend aufgefüllt. Zuerst die erste Etage und dann die zweite Etage.

Angefangen wird vom Zielpunkt aus. Zunächst wird der Zielpunkt als erstes Objekt in eine Liste mit Zielen eingespeist. Danach werden die umliegenden Felder aufgefüllt.

Bei dem Auffüllen wird zuerst geprüft, ob man bereits am Ziel ist. Wenn ja, dann wird das Auffüllen abgebrochen. Wenn nicht wird aufgefüllt.

Das Auffüllen funktioniert folgendermaßen:



Nach dem Auffüllen wird für jede Stelle an denen eine Zahl hinzugefügt wurde eine neue Koordinate gegeben, die in der nächsten Iteration abgefragt wird.

Dieses Prinzip zieht sich so lange weiter, bis das Ziel gefunden wurde. Dann wird derselbe Algorithmus für das zweite Stockwerk angewendet.

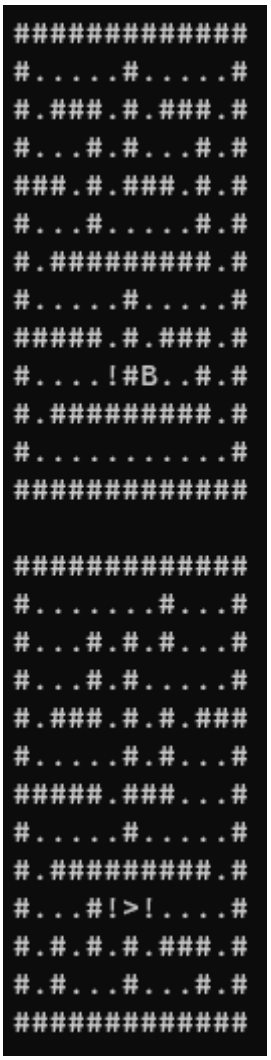
Lauf Algorithmus

Der lauf Algorithmus überprüft lediglich welche der benachbarten zahlen die kleinste ist und setzt in Richtung der kleinsten Zahl einen Pfeil, um damit anzuzeigen, dass das der richtige Weg ist.

Bei der unteren Etage wird zu der dort vorhandenen Zahl entweder eine 3 oder eine 8 dazu addiert.
Eine 3, wenn das Ziel sich bereits auf dem anderen Stockwerk befindet.
Eine 8, wenn das Ziel sich auf demselben Stockwerk befindet, weil man mindestens 2 schritte gehen muss und sich auch einmal runter und wieder hoch gehen muss, damit es sich lohnt.

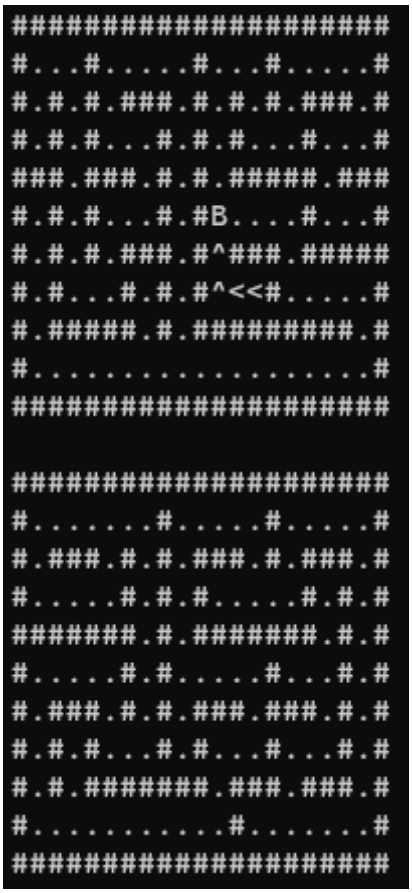
Beispiele

Zauberschule 0



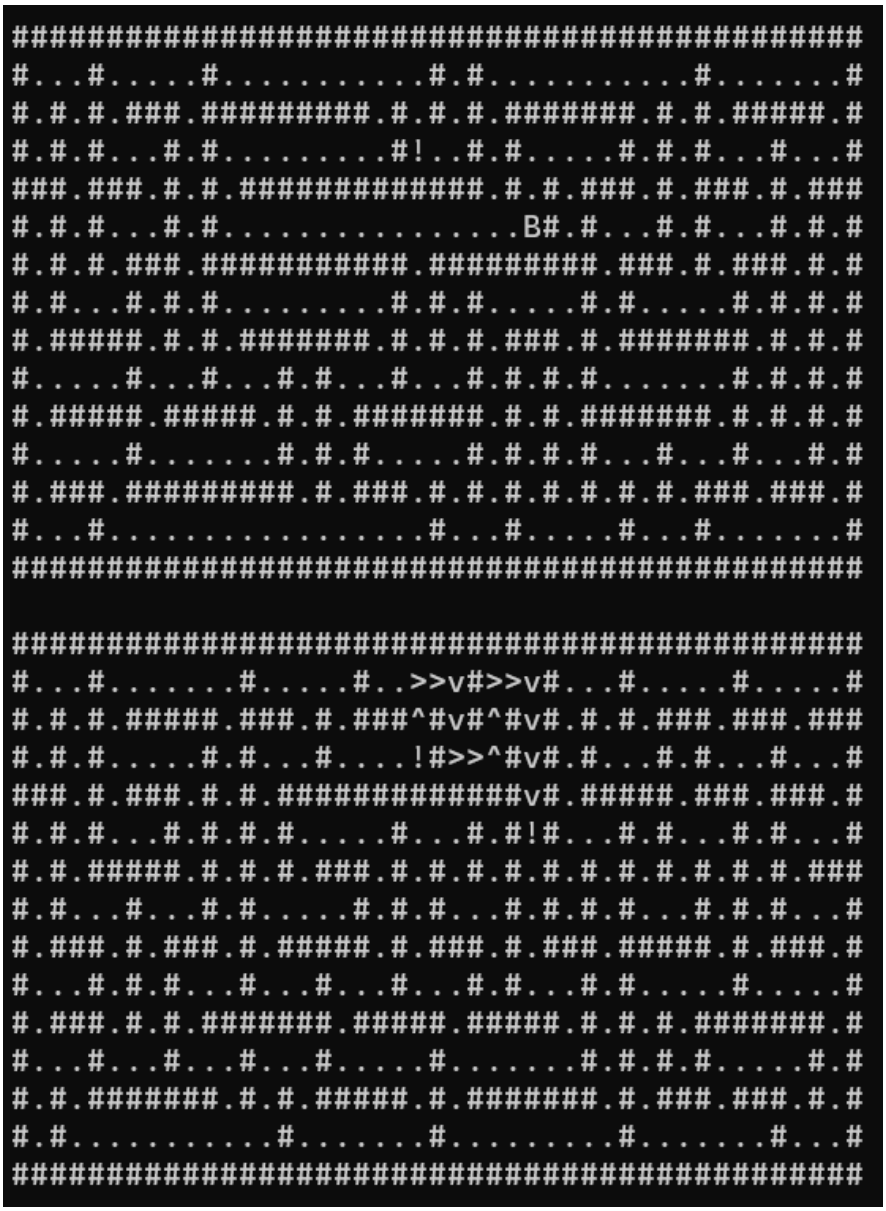
Hier sieht man, dass er den schnellsten Weg gefunden hat. Den Anfang A hat der Algorithmus gelöscht, weil er an der Stelle, an der er die Etage wechselt, auf beide stellen ein „!“ setzen soll, um klarzumachen, dass er hier gewechselt und angekommen ist.

Zauberschule 1



Bei dem Zweiten Beispiel findet der Algorithmus ebenfalls den schnellsten Weg, ohne sinnlos die Stockwerke zu wechseln.

Zauberschule 2.



In diesem Beispiel sieht man einen Fehler in meinem Lösungsansatz. Der Algorithmus findet zwar den Weg aber nicht den kürzesten. Dadurch dass der Flood Fill beide Etagen einzeln auffüllt und nicht alles auf einmal, werden die 2 Felder rechts vom Startpunkt übersprungen welche eigentlich Essenziell sind, um den schnellsten Weg zu finden.

Bei einem Zweiten Ansatz würde ich von dem Zielpunkt aus alles auffüllen. Bei den umliegenden Feldern + 1 und bei den Feldern auf dem anderen Stockwerk, wenn sie nicht belegt sind + 3. Den Lauf Algorithmus würde ich dann nur nach der kleineren Zahl suchen lassen in allen umliegenden Feldern und so den schnellsten Weg finden.

Weil mir allerdings nicht die Zeit reicht, das Projekt an dieser Stelle zu verbessern, reiche ich das in der Dokumentation beschriebene Projekt ein.

Quellcode

Program.cs

```
0 references
static void Main(string[] args)
{

    string pfadZurTxt = @"..\..\..\..\zauberschule0.txt";

    Schule schule = new(pfadZurTxt);

    Person person = new(schule);

    Ziel ziel = new(schule);

    FloodFill floodFill = new();

    schule = floodFill.AuffüllenDerStockwerke(schule, ziel, person);

    LaufAlgorhytmus lauf = new(schule, ziel, person);

    schule = lauf.SchnellstenWegFinden();

    Console.WriteLine("\n");

    schule.WriteSchule(schule);


}
```

Schule.cs

```
18 references
public class Schule
{
    public string[] linien;
    public string[,] grundrissInitErste;
    public string[,] grundrissInitZweite;
    public InitialisiereSchule initialisiere;

    19 references
    public Stockwerk ErsteEtage { get; set; }
    19 references
    public Stockwerk ZweiteEtage { get; set; }

    1 reference
    public Schule(string path)
    {
        linien = File.ReadAllLines(path);

        initialisiere = new InitialisiereSchule(linien);

        grundrissInitErste = new string[Convert.ToInt32(initialisiere.arrLänge), Convert.ToInt32(initialisiere.arrBreite)];
        grundrissInitZweite = new string[Convert.ToInt32(initialisiere.arrLänge), Convert.ToInt32(initialisiere.arrBreite)];

        ErsteEtage = new Stockwerk
        {
            Breite = Convert.ToInt32(initialisiere.arrBreite),
            Länge = Convert.ToInt32(initialisiere.arrLänge),
            Grundriss = initialisiere.OberesStockwerkAuslesen(grundrissInitErste, linien)
        };
        ErsteEtage.UrsprünglichePerson = initialisiere.UrsprünglichePersonAuslesen(ErsteEtage.Grundriss);
        ErsteEtage.UrsprünglichesZiel = initialisiere.UrschprünglichesZielAuslesen(ErsteEtage.Grundriss);

        ZweiteEtage = new Stockwerk
        {
            Breite = Convert.ToInt32(initialisiere.arrBreite),
            Länge= Convert.ToInt32(initialisiere.arrLänge),
            Grundriss = initialisiere.UnteresStockwerkAuslesen(grundrissInitZweite, linien)
        };
        ZweiteEtage.UrsprünglichePerson = initialisiere.UrsprünglichePersonAuslesen(ZweiteEtage.Grundriss);
        ZweiteEtage.UrsprünglichesZiel = initialisiere.UrschprünglichesZielAuslesen(ZweiteEtage.Grundriss);
    }
```

FloodFill.cs

```
2 references
public class FloodFill
{
    List<Ziel> Zwischenpunkte = new List<Ziel>();

    int floodNummer = 1;


    1 reference
    public Schule AuffüllenDerStockwerke(Schule schule, Ziel zielpunkt, Person person)
    {
        schule.ErsteEtage.Grundriss = AuffüllenDesStockwerks(schule.ErsteEtage, zielpunkt, person);

        schule.ZweiteEtage.Grundriss = AuffüllenDesStockwerks(schule.ZweiteEtage, zielpunkt, person);

        return schule;
    }
```

AuffüllenDesStockwerks Methode

```
2 references
private string[,] AuffüllenDesStockwerks(Stockwerk etage, Ziel zielpunkt, Person person)
{
    Zwischenpunkte.Add(zielpunkt);

    string[,] aktuelleEtage = etage.Grundriss;

    aktuelleEtage[zielpunkt.PositionX, zielpunkt.PositionY] = "B";
    aktuelleEtage[person.PositionX, person.PositionY] = "A";

    while (NotwendigkeitFürAuffüllungPrüfen(aktuelleEtage))
    {
        if (PrüfenObAmZiel(aktuelleEtage) == true)
        {
            Console.WriteLine("Ziel wurde gefunden.");
            break;
        }
        else
            FelderAuffüllen(aktuelleEtage);

        NeueKoordinatenHinzufügen(aktuelleEtage);

        //SinnloseKoordinatenLöschen(ersteEtage);

        floodNummer++;
    }

    ZielUndPersonPlatzhalterEntfernen(etage, zielpunkt, person, aktuelleEtage);

    Zwischenpunkte.RemoveRange(0, Zwischenpunkte.Count);
    floodNummer = 1;

    return aktuelleEtage;
}
```

Lauf Algorithmus

```
1 reference
public Schule SchnellstenWegFinden()
{
    _aktuelleEtage = SucheNachAktuelleEtage();

    int vordereZahl = SucheVordereZahl();
    int hintereZahl = SucheHintereZahl();
    int rechteZahl = SucheRechteZahl();
    int linkeZahl = SucheLinkeZahl();
    int andereEtageZahl = SucheAndereEtageZahl();

    while (!AbfragenObAmZiel())
    {


        if (_aktuelleEtage == Etage.Erste)
        {
            NächsterSchrittErste(vordereZahl, hintereZahl, rechteZahl, linkeZahl, andereEtageZahl);
        }
        else if (_aktuelleEtage == Etage.Zweite)
        {
            NächsterSchrittZweite(vordereZahl, hintereZahl, rechteZahl, linkeZahl, andereEtageZahl);
        }
        vordereZahl = SucheVordereZahl();
        hintereZahl = SucheHintereZahl();
        rechteZahl = SucheRechteZahl();
        linkeZahl = SucheLinkeZahl();
        andereEtageZahl = SucheAndereEtageZahl();
    }
```