Aufgabe 1

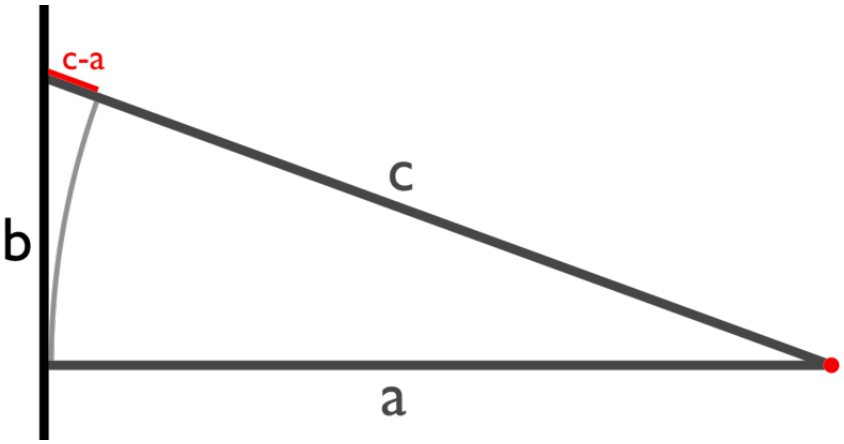
Nikolas Kilian

8. März 2019

1 Lösungsidee

Wenn es keine Hindernisse gibt, so ist der optimale Weg eine gerade Strecke vom Startpunkt zum Buspfad im 30° Winkel. Für Begründung davon siehe 1.1. Gibt es Hindernisse, so ist der optimale Weg der optimale Weg zu einem Eckpunkt, von dem die 30° Strecke offen ist, und dann diese 30° Strecke. Um das Optimum mit Hindernissen zu finden, muss man also alle Eckpunkte bestimmen, von denen aus diese 30° Strecke offen ist, und den optimalen Weg zu ihnen bestimmen. Unter den resultierenden Wegen ist das Optimum enthalten, also muss man nun nur noch die Zeit, zu der Lisa loslaufen muss, für alle Wege errechnen und den mit der spätesten Losgehzeit auswählen. Der optimale Weg zu diesen Eckpunkten lässt sich bestimmen mithilfe eines Sichtbarkeitsgraphen und Dijkstra's Algorithmus. Zum verhindern von Strecken durch unendlich dünne Wege (berührende Polygone) veränderet man den Sichtbarkeitsgraphen, sodass für jede normal sichtbare Linie nachträglich auf unendlich dünne Wege geprüft werden.

1.1 Berechnung



$$c = \sqrt{a^2 + b^2}$$

$$f := \frac{b}{a}$$

$$
\begin{aligned}
t(f) &= \frac{c-a}{v_{Lisa}} - \frac{b}{v_{Bus}} \\
&= \frac{\sqrt{a^2+b^2}-a}{v_{Lisa}} - \frac{af}{v_{Bus}} \\
&= \frac{\sqrt{a^2(1+f^2)}-a}{v_{Lisa}} - \frac{af}{v_{Bus}} \\
&= a\left(\frac{\sqrt{1+f^2}-1}{v_{Lisa}} - \frac{f}{v_{Bus}}\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{dt(f)}{df} &= \frac{da\left(\frac{\sqrt{1+f^2}-1}{v_{Lisa}} - \frac{f}{v_{Bus}}\right)}{df} \\
&= a\left(\frac{d\frac{\sqrt{1+f^2}-1}{v_{Lisa}}}{df} - \frac{d\frac{f}{v_{Bus}}}{df}\right) \\
&= a\left(\frac{\frac{d\sqrt{1+f^2}}{df}}{v_{Lisa}} - \frac{\frac{df}{df}}{v_{Bus}}\right) \\
&= a\left(\frac{\frac{1}{2\sqrt{1+f^2}} \cdot \frac{d1+f^2}{df}}{v_{Lisa}} - \frac{1}{v_{Bus}}\right) \\
&= a\left(\frac{f}{v_{Lisa}\sqrt{1+f^2}} - \frac{1}{v_{Bus}}\right)
\end{aligned}
$$

$$
\begin{aligned}
& \frac{dt(f)}{df} = 0 \\
\iff \quad & a\left(\frac{f}{v_{Lisa}\sqrt{1+f^2}} - \frac{1}{v_{Bus}}\right) = 0 \\
\iff \quad & a\frac{f}{v_{Lisa}\sqrt{1+f^2}} = a\frac{1}{v_{Bus}} \\
\iff \quad & \frac{f}{\sqrt{1+f^2}} = \frac{v_{Lisa}}{v_{Bus}} \\
\iff \quad & \left(\frac{f}{\sqrt{1+f^2}}\right)^2 = \left(\frac{v_{Lisa}}{v_{Bus}}\right)^2 \\
\iff \quad & \frac{f^2}{1+f^2} = \frac{v_{Lisa}^2}{v_{Bus}^2} \\
\iff \quad & \frac{1+f^2}{f^2} = \frac{v_{Bus}^2}{v_{Lisa}^2} \\
\iff \quad & \frac{1}{f^2} = \frac{v_{Bus}^2 - v_{Lisa}^2}{v_{Lisa}^2} \\
\iff \quad & f^2 = \frac{v_{Lisa}^2}{v_{Bus}^2 - v_{Lisa}^2} \\
\iff \quad & f = \sqrt{\frac{v_{Lisa}^2}{v_{Bus}^2 - v_{Lisa}^2}} \\
\iff \quad & f = \frac{v_{Lisa}}{\sqrt{v_{Bus}^2 - v_{Lisa}^2}}
\end{aligned}
$$

2 Umsetzung

Zur Umsetzung habe ich mich für eine Implementation in C# entschieden, mit einer Visualisierung mithilfe von WPF. Für die Generierung von Sichtbarkeitspolygonen verwende ich eine Implementation des Sweep-Line Algorithmus [Sources here]. Die Version des Algorithmus die ich verwende funktioniert wie folgt:

```
1  Let Intersections = Binary Search Tree, sorted by the order of
       intersection
2
3  foreach (Point p in Points sorted by their angle to Origin) {
4    Intersections.RemoveAll(Connected Edges on Clockwise Side of p);
5
6    if (IsVisible(p)) VisibleVertices.Add(p);
7
8    Intersections.AddAll(Connected Edges on Counterclockwise Side of p)
         ;
9  }
10
11 boolean IsVisible(p) {
12     if (!Origin.BetweenNeighbours(p) || !p.BetweenNeighbours(Origin))
           return false;
13     if (Origin and p are neighbours) return true;
14
15     if (Intersections is not empty and its leftmost element
          intersects the line from Origin to Target) return false;
16 }
```

P.BetweenNeighbours(A) gibt dabei zurück, ob für einen Punkt P der Teil eines Polygons ist ob A in dem in Abb. 1 grün markiertem Bereich liegt. Ist das Polygon in P konvex, so ist das Ergebnis immer false. Wenn der Rückgabewert dieser Methode false ist, so sind in einem reduziertem Sichtbarkeitsgraph die beiden Punkte nicht verbunden.

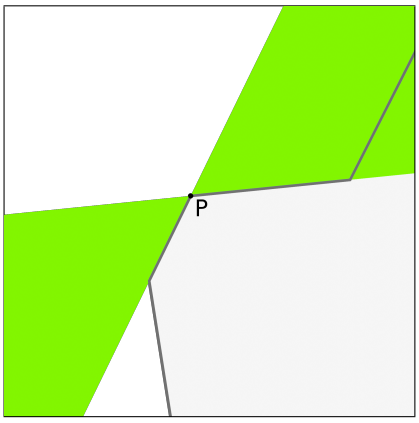


Abbildung 1: BetweenNeighbours

Um das durchgehen unendlich dünner Wege zu verhindern, speichere ich die hinzugefügten/entfernten Kanten, errechne die Strecke die sie auf der Strecke zum aktuellem Punkt einnehmen, und errechne die Überschneidungen der linken und rechten Seite.

Um nun ein reduzierten Sichtbarkeitsgraphen zu generieren muss dieser Algorithmus nun nur noch für alle Punkte ausgeführt werden.

Mit dem Sichtbarkeitsgraphen fertig genreriere ich nun eine Heuristik mit Dijkstras Algorithmus, jedoch generiere ich diese nur bis allen Endpunkten (Enden der 30° Strecken, auf dem Buspfad) von Dijkstra besucht wurden (/an der Spitze der Priotitätsliste waren).

Da Dijkstra's Algorithmus nicht immer alle Knoten besucht, muss der Sichtbarkeitsgraph auch nicht vollständig generiert werden. Um dies auszunutzen berechne ich das Sichtbarkeitspolygon nur für Punkte die Dijkstra besucht.

Sobald die Heuristik fertig generiert ist, errechne mit dieser die optimale Strecke zu allen Endpunkten und die Zeit die Lisa braucht um diese abzulaufen, und die Zeit die der Bus braucht, um dorthin zu kommen. Damit errechne ich die Zeit zu der Lisa losgehen muss für alle diese Wege, vergleiche diese und nehme den Weg mit der spätesten Startezeit. Dieser Weg ist der optimale Weg, und somit das Ergebnis.

3 Beispiele

4 Code