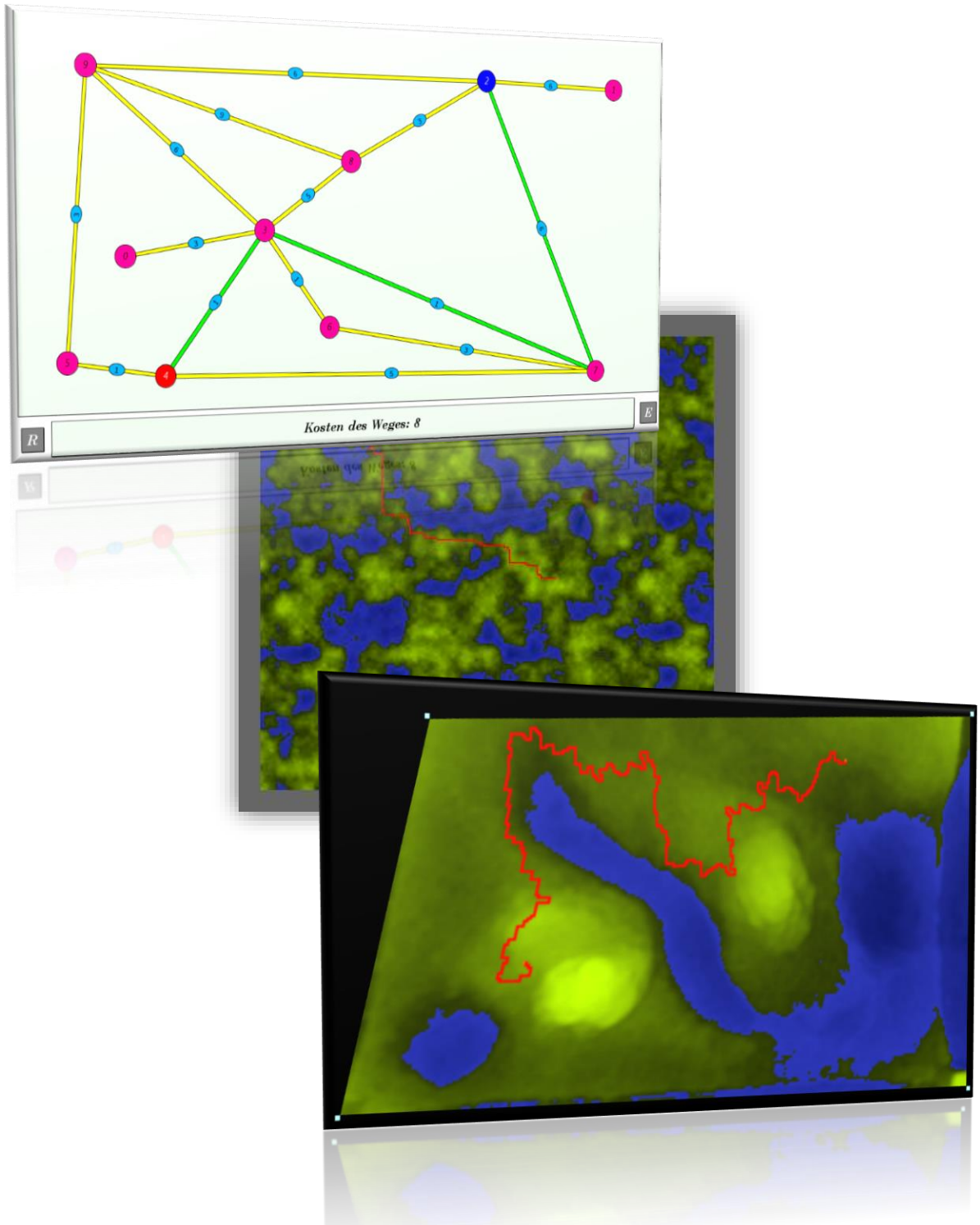
1. OKTOBER 2014



Informatikprojekt 1

Pathfinding

Ein Projekt von Tobias Wagner, Simon Jacobs und Jonas Voelcker

Inhaltsverzeichnis

1. Projektverlauf

1.1 Unser Vorgehen

Das Projekt begann mit dem Kennenlernen der Projektteilnehmer und der anschließenden Aufteilung in Gruppen.

Daraufhin hat sich unsere Gruppe über Facebook und WhatsApp organisiert und abgesprochen, wann und wo Gruppentreffen stattfinden sollten.

Bei unseren ersten Gruppentreffen haben wir geklärt, wer welche Aufgaben übernehmen sollte, bis wann diese fertig sein sollen und haben darüber Protokoll geführt.

Aufgrund einiger zeitlicher Probleme wurde im zweiten Anlauf mit Deadlines und aufeinanderfolgenden Aufgaben gearbeitet.

Im späteren Projektverlauf stellten wir fest, dass eine derart genaue Planung nicht zielführend war. Daher wurde ab sofort immer öfter zusammen an den Lösungen gearbeitet, um konzentrierter und koordinierter eine Lösung zu finden.

Nach der Vorstellung des Sandkastens wurden viele der Arbeiten im Büro durchgeführt, denn ohne die Kinect konnte das Programm nicht adäquat getestet werden.

1.2 Lerninhalte

1.2.1 Basiswissen

In den ersten Treffen wurden uns der Umgang mit Processing und die spezielle Syntax dieser Java-Erweiterung beigebracht.

Es gab zudem eine Einführung in das Thema Graphen inkl. dem Königsberger Brückenproblem, gerichteten und ungerichteten Graphen, Knoten und Kanten sowie der Kantenbewertung.

1.2.2 Dijkstra-Algorithmus

Der Algorithmus von Dijkstra dient der Findung des kürzesten Weges in einem Graphen mit positiven Kantengewichten.

Er startet an einem bestimmten Knoten mit seiner Suche und schaut in alle möglichen Richtungen, merkt sich dabei die Kosten und Folgeknoten und speichert sie in einer Liste.

Dann folgt er in jedem Schritt dem Weg mit den bisher geringsten Gesamtkosten, abermals in jede Richtung, aber nur zu Knoten, die noch nie besucht worden sind.

So findet der Algorithmus irgendwann den gewünschten Knoten. Aufgrund des Besuchens des stets kürzesten Einzelweges ist dieses erste Erreichen des Zielknotens automatisch der kürzeste Weg.

Findet der Algorithmus irgendwann keinen Folgeknoten mehr, der noch nicht besucht worden ist und war darunter der gesuchte Knoten nicht aufgelistet, so gibt es zwischen den beiden Knoten keinen Weg.

1.2.3 A*-Algorithmus

Der A* ist ein sogenannter informierter Suchalgorithmus, er besitzt also Zusatzinformationen über die mögliche Entfernung zum Ziel, diese Information nennt sich Heuristik.

Die geschätzten Kosten des Weges von einem Knoten zum Ziel setzen sich aus zwei Teilen zusammen:

- g(x) sind die Kosten vom Startknoten zum Knoten x, diese betragen beim Startknoten selbst 0 und werden im weiteren Verlauf für jeden Knoten neu festgelegt, indem sich bei Erkunden eines Knotens immer der jeweilige kürzere Weg gemerkt wird.
- h(x) ist der Wert, den die Heuristik ergibt, dies kann beispielsweise die Luftlinie, Manhattan-Distanz (x-Differenz + y-Differenz) oder eine andere grobe Schätzung sein. Diese darf jedoch den tatsächlichen Weg niemals überschätzen!

Wie beim Dijkstra werden von jedem Knoten aus alle Nachfolger notiert, doch diesmal entscheidet nicht der bisher zurückgelegte Weg sondern die Summe aus bisherigem Weg und Heuristik über den nächsten betrachteten Knoten.

Hierdurch wird sichergestellt, dass der Algorithmus nicht in die entgegengesetzte Richtung losläuft und erst später die richtige Richtung findet.

1.3 Schwierigkeiten

1.3.1 Absprung zweier Teilnehmer

Unser zentrales Problem bestand darin, dass wir ursprünglich mit 5 Teilnehmern gestartet sind, sich jedoch nach wenigen Wochen die erste Teilnehmerin aus persönlichen Gründen aus der Gruppe verabschieden musste.

Ein weiteres Gruppenmitglied gab kaum einen Beitrag zu unserer Arbeit und verabschiedete sich letztendlich auch.

Dieses Fehlen an Tatkraft mussten wir Verbliebenen kompensieren. Bemerkenswerterweise war jedoch die Absprache in dieser kleineren Gruppe deutlich einfacher.

1.3.2 Absprache in der Gruppe

Gerade zu Beginn des Projektes fiel die Einhaltung von Fristen sehr schwer, teilweise wurden Arbeiten so spät angefangen, dass nur noch nüchtern festgestellt werden konnte, dass die Teilaufgabe zu schwierig für die betreffende Person war und deswegen eine längere Auseinandersetzung mit dem Thema nötig gewesen wäre.

Auch später war es schwierig, sich auf einen gemeinsamen Termin zu einigen, da wir verschiedene Arbeitszeiten, Übungsgruppen und Anfahrtswege haben. Somit war auch das spontane Treffen selten möglich.

1.3.3 Rekursion oder Iteration?

Der erste A*-Algorithmus, den wir probiert haben, rief die Methode noch rekursiv auf, dies führte bei längeren Wegen unweigerlich zu einem StackOverflowError, da sich das Programm alle lokalen Variablen der aufrufenden Methoden merken musste.

Durch eine fehlerhafte Kommunikation seitens des Programmierers konnte dieses Problem erst nach vielen Tagen aus dem Weg geräumt werden, als dann die rekursive Vorgehensweise in eine einfache do-while-Schleife umprogrammiert wurde.

1.3.4 Der Sandkasten-Stream

Leider gab es auch auf Seiten der Sandkasten-Crew Organisationsprobleme, da ein Teil dieses Teams in Hongkong war und sich daher die Kommunikation schwierig gestaltete.

Zuerst lief der Stream unsauber, später wurden noch Änderungen durchgeführt, die uns die Arbeit erleichterten, dies hat aber teils viel Zeit gekostet, weswegen wir nicht direkt durchprogrammieren konnten.

In zukünftigen Projekten wäre es wünschenswert, aufeinander aufbauende Projekte in verschiedenen Semestern anzusiedeln, damit sich zeitliche Probleme nicht auf anderer Leute Arbeit auswirkt.

1.4 Organisation und Meilensteine

Für die Programmierung des ersten Aufgabenteils (Graphen-Darstellung) sahen wir folgende 5 Gebiete vor:

1. Programmlogik (Klassen Graph, Node und Edge)
2. Objekterzeugung (Für die Erstellung eines Anfangsgraphen)
3. Zeichnen des Graphen
4. Interaktion (Highlights für ausgewählte und anvisierte Elemente, Verschiebung, etc.)
5. Dokumentation des Ergebnisses.

Nach diesen ersten Vorbereitungen wurden abermals Zuständigkeiten vergeben, dieses Mal jedoch mit konkreten Zeitfenstern und der Beachtung aufeinander aufbauender Aufgaben:

1. Logik (11.04. - 13.04.)
2. Darstellung (14.04. - 21.04.)
3. Eventhandler (14.04. - 21.04.)
4. Recherche über Wegfindungsalgorithmen + Aufbereitung (11.04. - 28.04.)
5. Dokumentation (22.04. - 28.04.)

Der zweite Aufgabenteil (Wegfindung auf der 2D-Karte) war eine etwas aufwendigere Erweiterung des ersten Teils, nach knapp 2 Wochen stand die Lösung dank einiger gemeinsamer Programmiersessions.

Nach der Vorstellung des Sandkastens inkl. Kinect 2 und Beamer mit zur Verfügung gestelltem Stream ging der Hauptanteil des Projektes los.

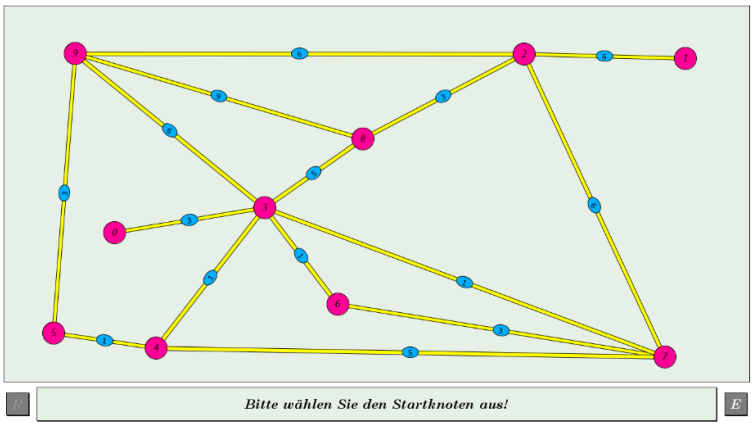
Das Interface wurde an die neuen Gegebenheiten angepasst. Eine frei transformierbare Ausgabe mit 4 Ankerpunkten, ein Auswahlmodus für die Start- und Zielknoten und keinerlei Textausgaben, da ja eine Ausgabe auf dem Sandkasten stattfinden sollte.

Der Stream musste korrekt ausgelesen werden und in ein gültiges Objekt der Klasse Grid überführt werden.

Das fertige Programm ist nun in der Lage, mit dem bereitgestellten Stream in Echtzeit den Sandkasten auszulesen, Wasser und Land zu simulieren und gefundene Wege auf dem Sand anzuzeigen, dies in verschiedenen Schwierigkeitsgeraden.

2 Die Ergebnisse

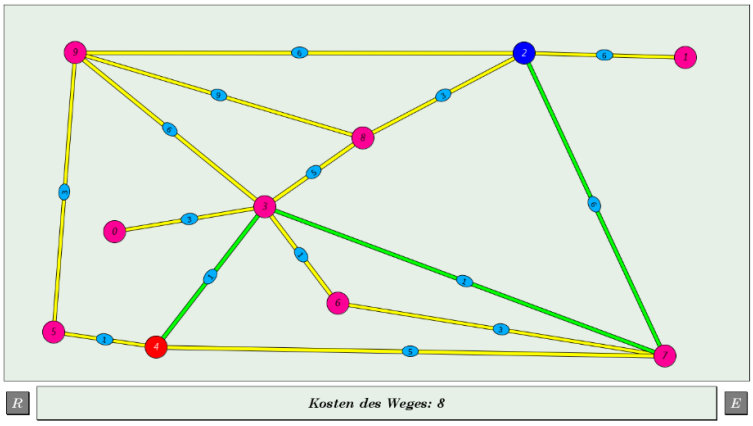
2.1 Graphen und der Dijkstra-Algorithmus



Dieses erste Programm zeigt einen beliebigen ungerichteten Graphen mit Kantenbewertungen an.

Man kann Knoten hinzufügen und löschen, Kanten neu verknüpfen oder entfernen und die Knoten auf dem Raster frei verschieben. Zudem wird im unteren Teil eine Statusmeldung angezeigt, um dem Nutzer die Bedienung zu erleichtern.

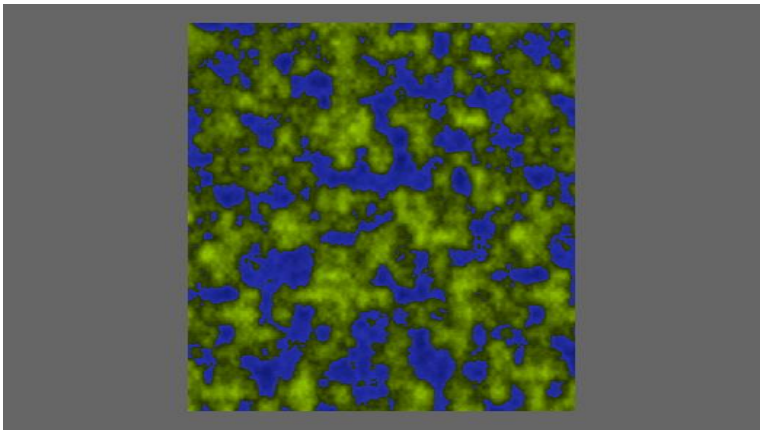
Alle erarbeiteten Ergebnisse lassen sich in einem speziellen Dateiformat (.pathf) speichern und auch wieder abrufen.



Durch Selektion eines Start- und Zielknotens berechnet der Dijkstra-Algorithmus den kürzesten Weg zum Ziel und zeigt diesen grün an. Der Startknoten wird blau und der Zielknoten rot dargestellt.

Zusätzlich werden die Kosten des Weges direkt summiert und dem Nutzer angezeigt.

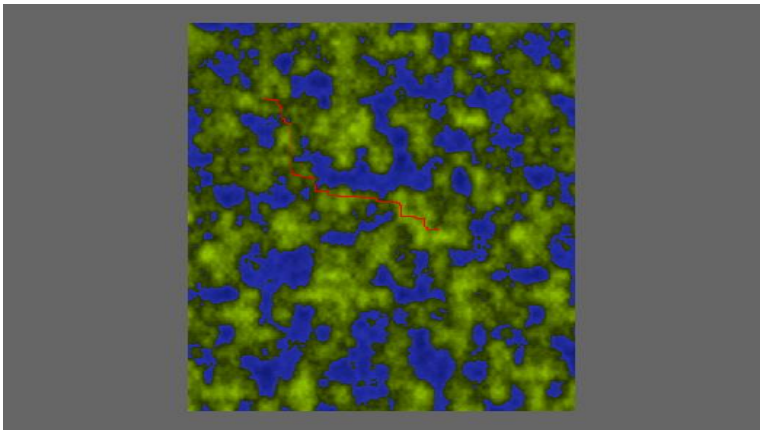
2.2 Wegfindung auf der 2D-Karte



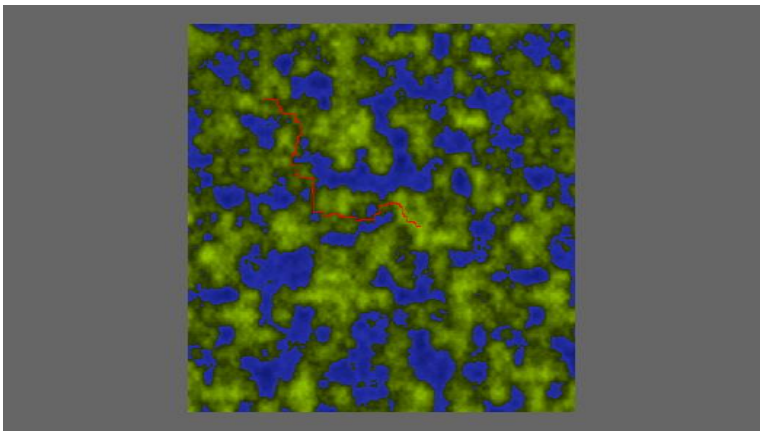
Als zweite Aufgabe sollte eine Wegfindung auf einer zweidimensionalen Karte, welche für jeden Punkt eine spezifische Höhe besitzt, realisiert werden.

Die Höhen der Punkte liegen zwischen 0 und 1. Ein Wasserlevel gibt an, welcher Teil der Punkte überflutet ist. Der Rest ist begehbar und steht der Wegsuche zur Verfügung.

Zur leichteren Sichtbarkeit der Höhen wird zwischen hellgrün (hoch - Land), dunkelgrün (tief - Land), hellblau (seicht - Wasser) und dunkelblau (tief - Wasser) unterschieden.



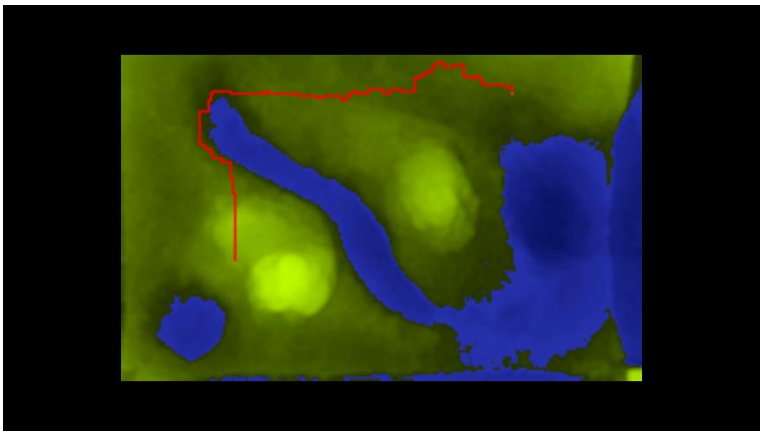
Wählt man nun einen Start- und Zielknoten aus, wird mithilfe des A*-Algorithmus der kürzeste Weg berechnet und daraufhin angezeigt. Die Start- und Zielknoten werden orange dargestellt, während der Weg rot erscheint.



Zudem lässt sich die Schwierigkeit des Terrains anpassen, um verschiedene Fortbewegungsarten zu simulieren, hier sieht man Stufe 9, während zuvor die Stufe 1 gezeigt worden ist.

Es ist gut zu sehen, dass sich der Pfad hier eher an der Küste entlang bewegt und nur in Ausnahmefällen die Berge erklommen werden.

2.3 Wegfindung im Sandkasten (mit Kinect 2 und Beamer)



Die Verbindung aus unserem Sandkasten, einer Kinect 2 samt Stream und dem Beamer macht aus unserem Programm ein interaktives Wegfindungserlebnis.

In Echtzeit werden auf dem Sandkasten die Höhen und Tiefen farbig eingeblendet und jederzeit lässt sich die Aktualisierung stoppen, um einen Weg im aktuellen Bild zu suchen.

Man muss dafür einfach nur das Keystoning abschalten und kann dann Start- und Zielpunkt auswählen. Anschließend kann man das Keystoning wieder aktivieren und das Ergebnis betrachten.

Das Programm nutzt hierfür weiterhin den A*-Algorithmus nur wird ständig eine neue Instanz der Karte erzeugt, auf der gearbeitet werden kann.

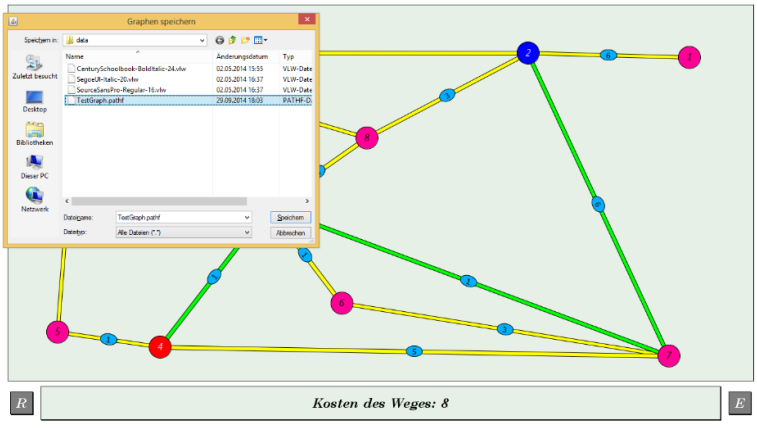
3 Bedienung der Programme

3.1 Graphen und der Dijkstra-Algorithmus

Auf dem Bild sieht man die Bedienhilfe, die sich mit Druck auf die Taste H öffnet.

Der Graph lässt sich durch +, - und * kombiniert mit einer Zahlentaste erweitern und ändern, das + erzeugt einen neuen Knoten an der Mausposition, während das - den selektierten Knoten samt aller Verbindungen löscht.

Mit der Taste * lässt sich ein bestehender Knoten markieren, geht man nun mit der Maus über einen anderen Knoten und drückt eine Taste von 1–9, so wird die entsprechende Verbindung erzeugt. Die 0 entfernt eine etwaige Verbindung.



Zudem lässt sich ein Graph über die Taste S speichern (siehe Bild), hierbei muss die Endung zwingend .pathf sein, alle anderen Endungen lehnt das Programm ab.

Außerdem kann ein bereits gesicherter Graph auch wieder durch das Drücken von L geladen werden.

Per Rechtsklick und Halten lassen sich die Knoten auf dem Raster frei bewegen.

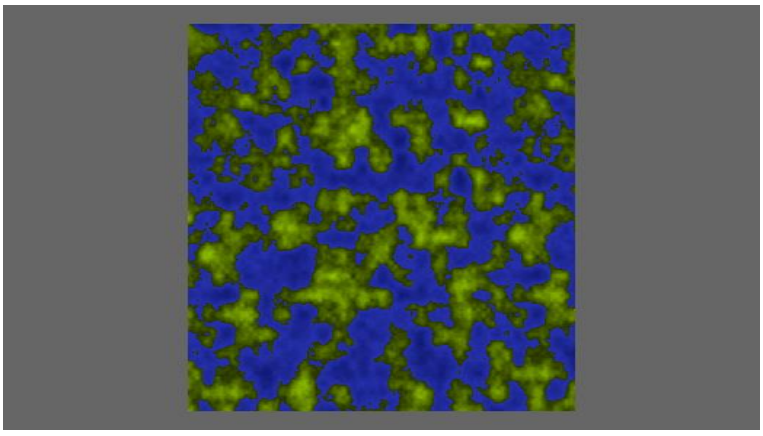
Abschließend kann man mit der linken Maustaste nacheinander den Start- und Zielknoten auswählen, der Weg wird berechnet sowie angezeigt (falls vorhanden) und es erscheint eine passende Statusmeldung. Mit der Taste R lässt sich die Auswahl zurücksetzen.

3.2 Wegfindung auf der 2D-Karte

Auch in diesem Programm erreicht man die Bedienhilfe mit der Taste H.

Um verschiedene Fortbewegungsarten zu simulieren, lässt sich mit den Tasten 0-9 die Terrainschwierigkeit ändern. 0 bedeutet, dass alle Schritte gleich viel Kosten, je höher die Schwierigkeit, desto teurer ist es, Berge zu erklimmen oder in Täler zu reisen.

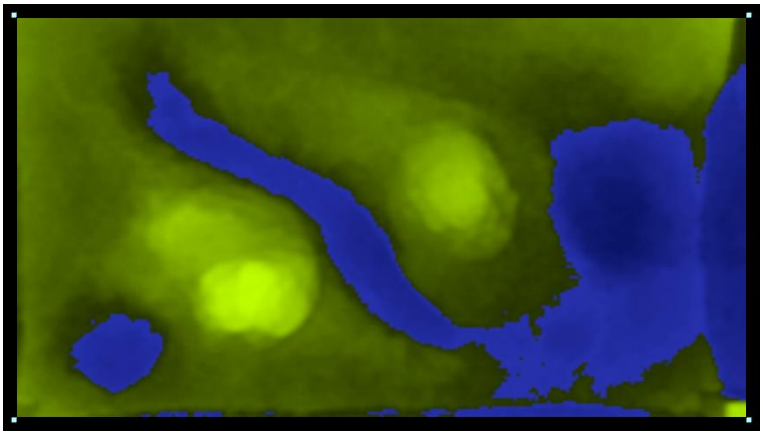
Die Taste L dient dem Laden fertiger Karten.



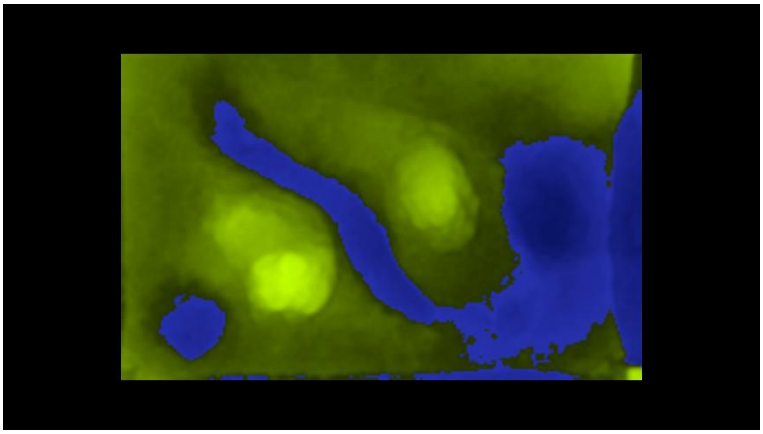
Auf diesem Bild sieht man die Anpassung des Wasserlevels mittels Mausrad.

3.3 Wegfindung im Sandkasten (mit Kinect 2 und Beamer)

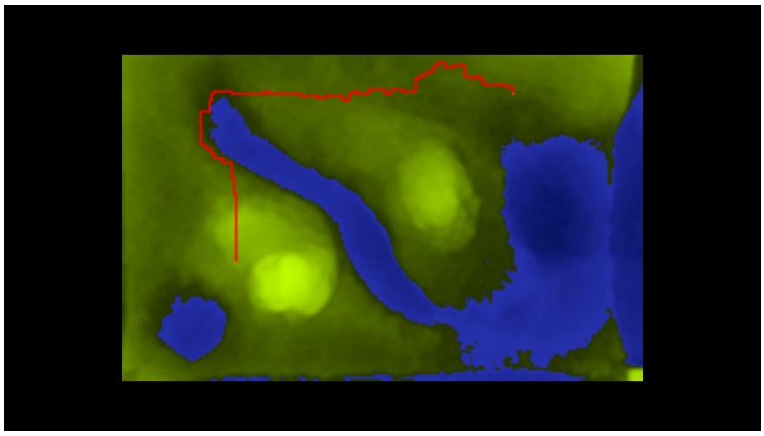
Dieses Programm funktioniert nur in Verbindung mit einem bereits eingerichteten Stream, welcher die Höhen des Sandkastens übertragen soll.



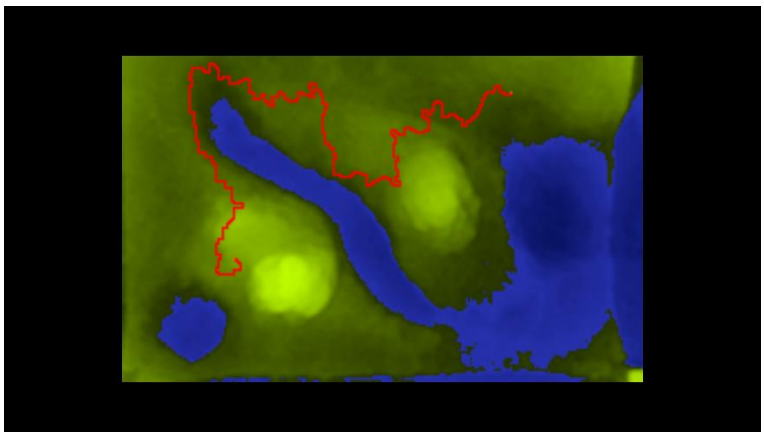
Ist einmal die Verbindung hergestellt, sieht man ein Bild, das dem oberen ähnelt. Die vier Kanten sind in den oberen vier Ecken platziert, damit man möglichst viel sieht.



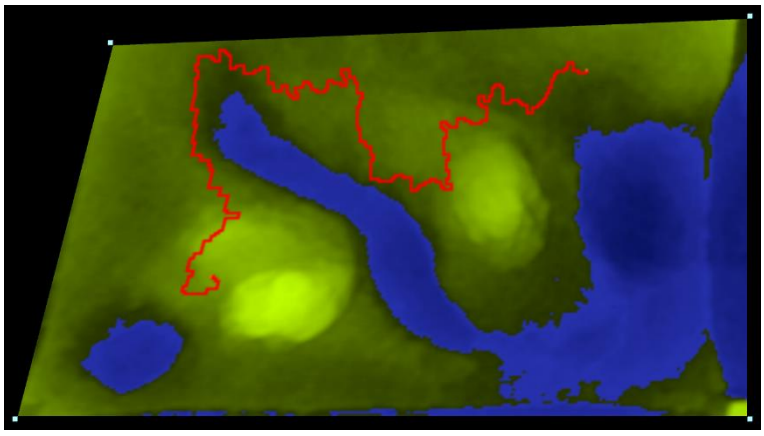
Möchte man nun einen Weg finden, so ist es zuerst einmal wichtig, durch die Taste K das Keystoning zu deaktivieren, ist einem die angezeigte Karte zu groß, kann sie mit der Taste – verkleinert werden, mit + wird sie vergrößert.



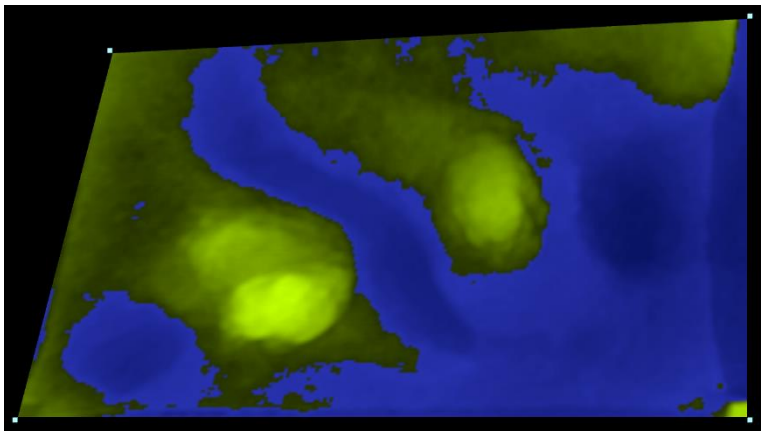
Wählt man nun nacheinander einen Start- und Zielknoten aus, berechnet das Programm den kürzesten Weg und zeichnet diesen ein.



Mit Hilfe der Tasten 0-9 lässt sich auch hier die Terrainschwierigkeit kalibrieren, wie das obige Beispiel gut zeigt.



Zurück in der Ansicht mit Keystoning (durch abermaligen Druck der Taste K) lässt sich per Drag&Drop jeder einzelne Eckpunkt verschieben, diese sollte man so legen, dass die Berge und Täler im Sandkasten korrekt angezeigt werden.



Wie auch bei der Vorgängerversion lässt sich weiterhin mit dem Mausrad der Wasserpegel einstellen.

4. Fazit

4.1 Tobias Wagner

Für mich war das Projekt eine super Gelegenheit, um positive und negative Erfahrungen zu sammeln, die während einer Teamarbeit auftreten können.
Zum einen erinnere ich mich gerne an die gute Stimmung innerhalb des Teams zurück. Man ist super untereinander ausgekommen und persönliche Streitigkeiten gab es nicht.
Diese Tatsache möchte ich aus meiner persönlichen Sicht besonders hervorheben, denn nichts ist wichtiger, als dass man in einem Team von Menschen umgeben ist, bei denen die Chemie und die Kommunikation untereinander funktioniert.

Dies wirkte sich auch positiv auf den Entwicklungsprozess aus. Jeder wusste sich mit seinen Stärken im Team einzubringen und verhalf der Teamarbeit so zu guten Fortschritten.

Zwischenzeitlich musste leider auch die Erfahrung gemacht werden, dass aus wichtigen Gründen manche das Team verlassen mussten und dass die noch anfallende Arbeit auf den Rest des Teams umverteilt werden musste.

Ein wenig störend, aber bestimmt nicht ohne Grund geplant, empfand ich die Miteinbeziehung eines Studenten, der ein paar Wochen nicht erreichbar war und somit das Projekt zeitlich im Stillstand war.

Andere technische Probleme sind in einem experimentellen Projekt unausweichlich und wurden super gelöst.

Jederzeit würde ich an einem derartigen Projekt wieder mitwirken und meinen Teil dazu beitragen, es erfolgreich zu beenden.

4.2 Simon Jacobs

Zum einen nehme ich aus dem Projekt mit, dass konkrete Absprachen wichtig sind, um den Projektverlauf besser zu strukturieren.

Außerdem ist ein genauer Abgabetermin wichtig, um zu verhindern, dass ein Projekt unnötig in die Länge gezogen wird.

Auch eine genaue Festlegung, wer für welchen Aufgabenteil verantwortlich ist, ist notwendig, um jeden mit einzubinden.

4.3 Jonas Voelcker

Dieses Projekt zeigte mir, dass man in der Zusammenarbeit mit anderen viele Kompromisse eingehen muss. Es darf nicht immer alles in den eigenen Augen perfekt sein, denn so unterdrückt man die Individualität der Mitstreiter.

Wenn die Aufgaben bereits verteilt sind, sollte man so geduldig sein, auf den anderen zu warten, um anschließend das Ergebnis aus den Einzelteilen zusammenzusetzen. Das erhöht die eigene Frustrationstoleranz und erhöht die Motivation der Anderen.

Für die Zukunft weiß ich eines ganz genau: Ich muss mich der Gruppe unterordnen und nicht die Gruppenmitglieder mir. Allein diese Erkenntnis war das Projekt schon wert.

Glücklicherweise ist dies jedoch nicht alles, was ich mitnehmen konnte, denn mit Processing haben wir ein klasse Programm kennengelernt, um mal eben schnell etwas auf simpelste Art und Weise zu testen, ich hoffe sehr, das im späteren Leben sinnvoll nutzen zu können.