|  |
| --- |
| Elmer Lukas, Heidt Christina, Treichler Delia  10. Dezember 2011 |

|  |
| --- |
| Studienarbeit |
| 3D |
|  |

****

# Dokumentinformationen

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 01.12.2011 | 1.0 | Erste Version des Dokuments | lelmer |
| 11.12.2011 | 1.1 | Weiter Dokumentiert, Bilder eingefügt | lelmer |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc311381292)

[1.1 Änderungsgeschichte 1](#_Toc311381293)

[1.2 Inhaltsverzeichnis 1](#_Toc311381294)

[2 Übersicht 2](#_Toc311381295)

[2.1 Problem 2](#_Toc311381296)

[2.2 Idee 2](#_Toc311381297)

[3 Ansatz 2](#_Toc311381298)

[3.1 Genereller Ansatz 2](#_Toc311381299)

[3.2 Ausrichtung 2](#_Toc311381300)

[3.3 Anzahl Elemente und Krümmungen 3](#_Toc311381301)

[3.4 Ausblenden zu kleiner Project Notes 3](#_Toc311381302)

[4 Umsetzung 4](#_Toc311381303)

[4.1 Mathematische Grundlagen 4](#_Toc311381304)

[4.2 Programmierung 4](#_Toc311381305)

[4.2.1 Implementation 4](#_Toc311381306)

[4.2.2 Resultat 5](#_Toc311381307)

[5 Weiterentwicklung 6](#_Toc311381308)

[6 Fazit 6](#_Toc311381309)

# Übersicht

## Problem

In der Übersicht bei den Project Notes muss erkannt werden, dass nach unten oder oben gescrollt werden kann. Zusätzlich wäre es schön, wenn man die Anzahl Project Notes einfach erkennen und abschätzen könnte.

## Idee

Durch eine 3D Ansicht der Project Notes kann durch den 3D Effekt erkannt werden, dass im Hintergrund noch weitere Project Notes existieren. Da die Project Notes im Hintergrund kleiner werden, kann auch die Anzahl ungefähr abgeschätzt werden, wobei man die fokussierten Project Notes immer noch grösser darstellt.

# Ansatz

Um eine Lösung zu erarbeitet, wurde zuerst ein sehr genereller Ansatz gesucht. Dieser wurde dann über mehrere Schritte verfeinert.

## Genereller Ansatz

Als Grundlage wurde die Perspective Wall [chi91] genommen. Das sieht grundsätzlich folgendermassen aus:

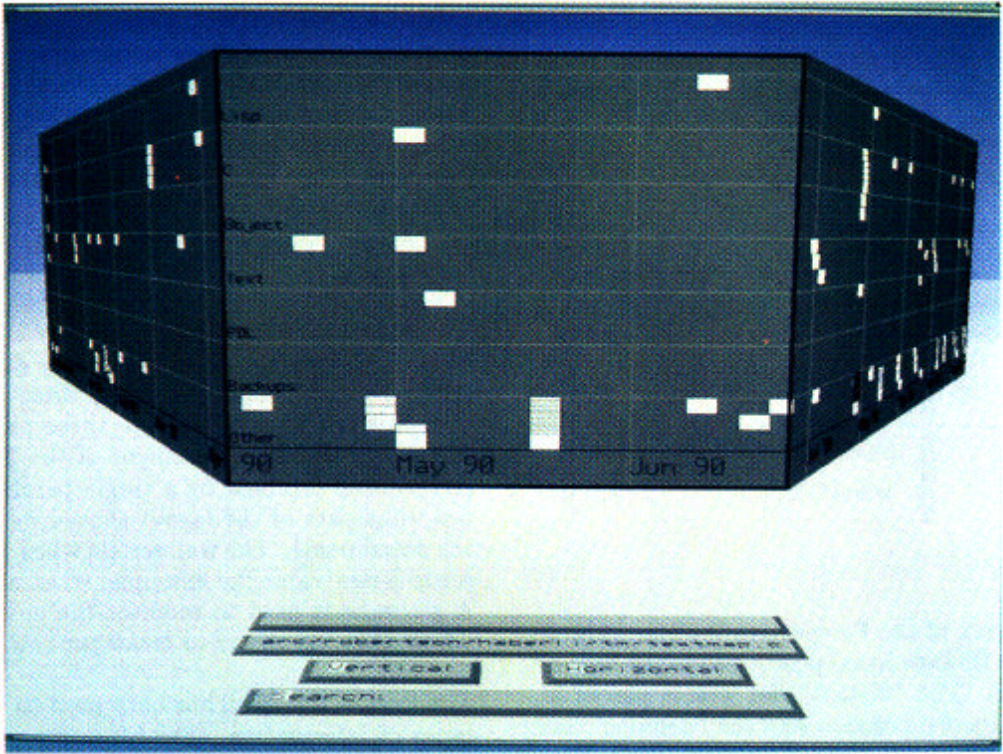


Abbildung 1 - Beispiel einer Perspective Wall, [chi91]

## Ausrichtung

Da die Project Notes von oben nach unten und nicht von links nach rechts gescrollt werden sollen, wurde in einem ersten Schritt die Wall um 90 Grad gedreht. Das sieht dann folgendermassen aus:

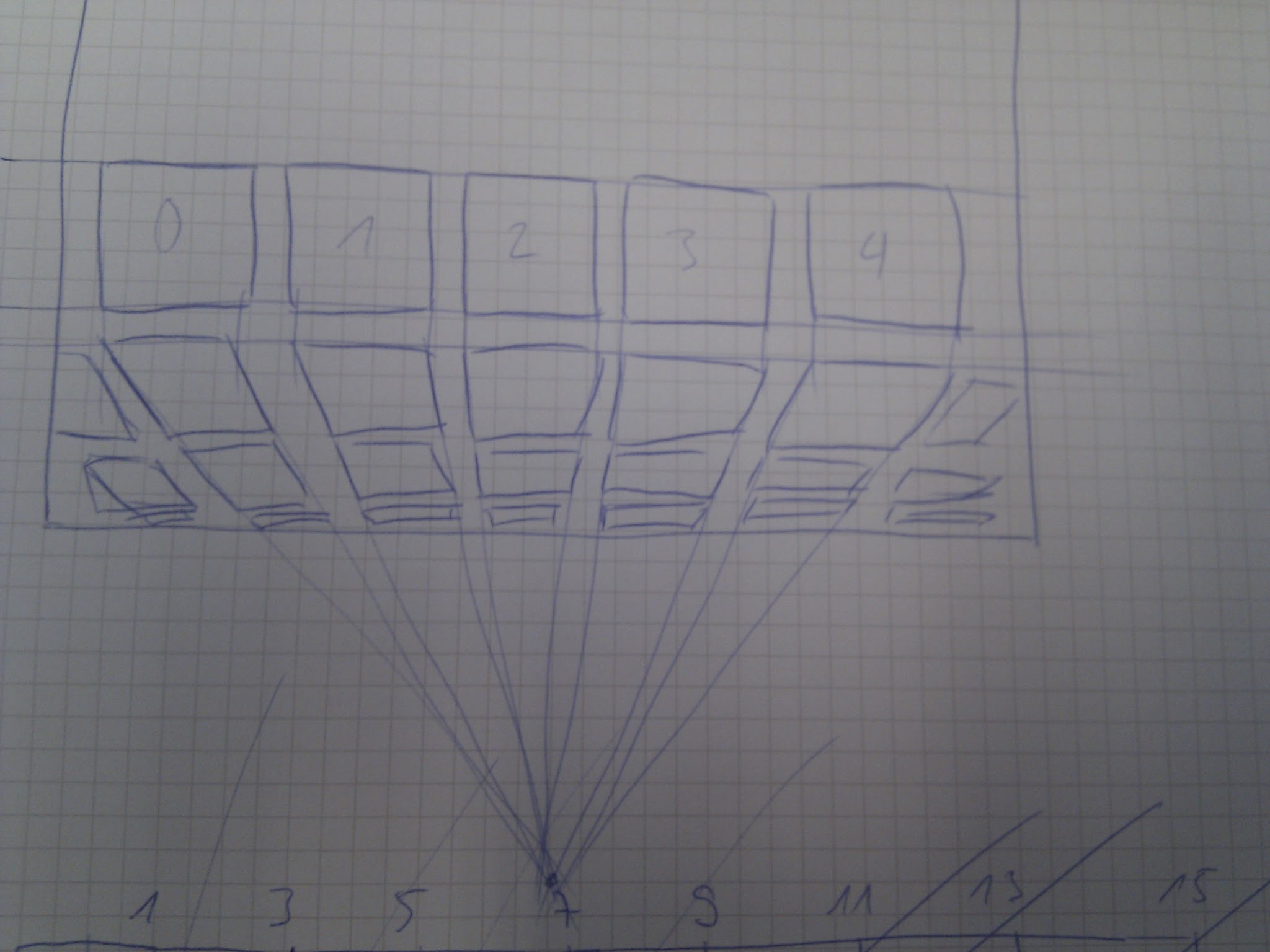


Abbildung 2 - Skizze 3D ohne Krümmung, 90 Grad gedreht

## Anzahl Elemente und Krümmungen

Die originale Perspective Wall beschränkt sich auf drei verschiedene Spalten: die mittlere Spalte ist flach dargestellt, während die Spalte rechts und links beide perspektivisch verzogen sind.

Da mehr als nur drei Reihen Project Notes dargestellt werden sollen, wurden noch weitere Reihen hinzugefügt, und die Neigung der perspektivischen Verzerrungen angepasst. Geometrisch bedeutet das, dass die Oberfläche der abstrakten 3D Figur von einer flachen Oberfläche zu einer runden Oberfläche wird. Das wiederum heisst, dass die Linien nicht mehr gerade sondern neu rund sind. Folgende Skizze zeigt ein Beispiel auf:

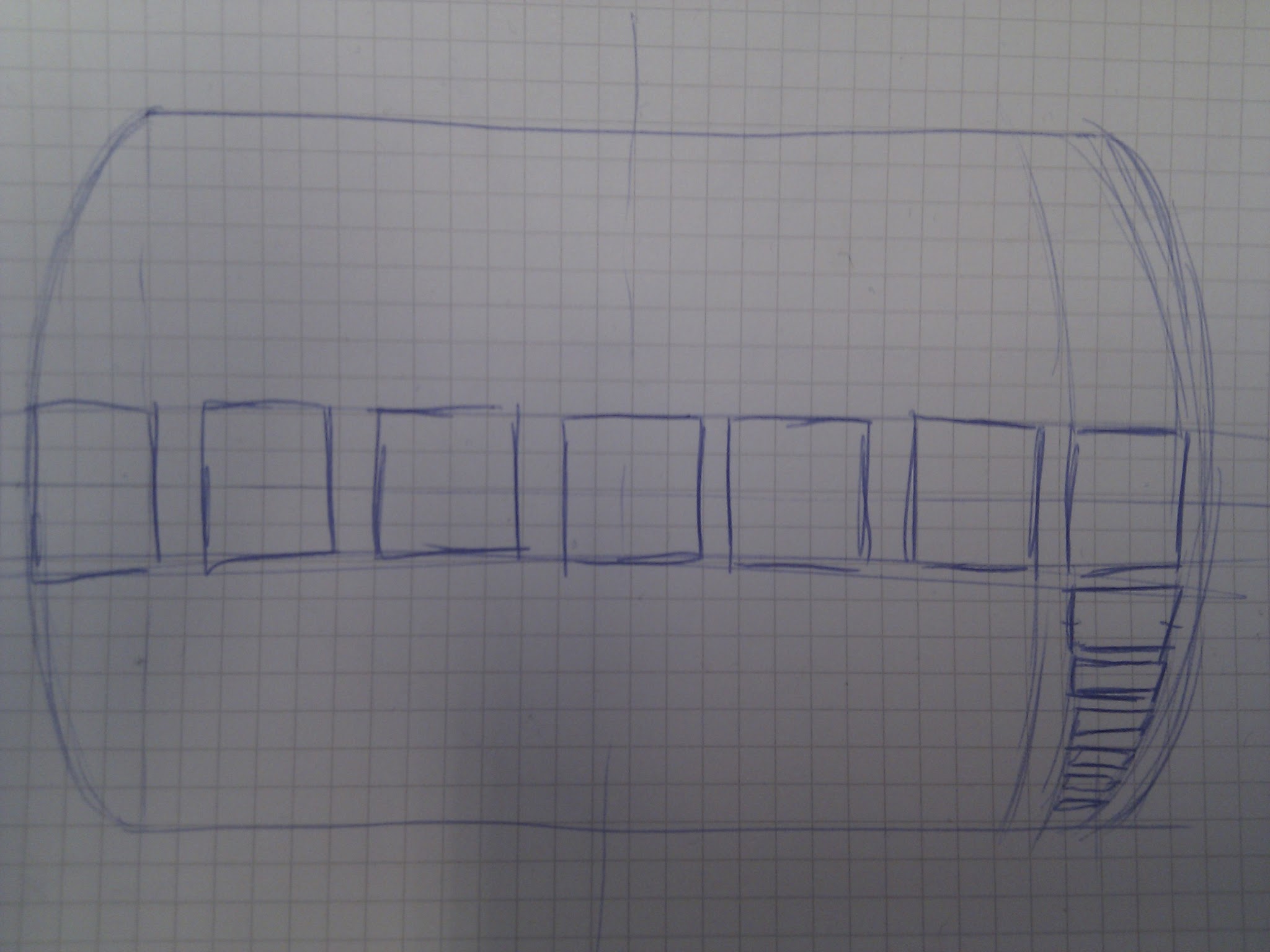


Abbildung 3 - Skizze 3D mit Krümmung

## Ausblenden zu kleiner Project Notes

Damit am unteren Rand der Darstellung nicht zu stark gestauchte Project Notes vorkommen, müssen einige Elemente unten (3D: hinten) ausgeblendet werden. Dies kann z.B. so gemacht werden, dass alle Project Notes, die weniger hoch als 5 Pixel hoch sind, ausgeblendet werden.

# Umsetzung

## Mathematische Grundlagen

Grundsätzlich benötigt man eine Funktion mit zwei Parametern:

X: die Position der X-Achse

Y: die Höhe, auf der die Project Note gezeichnet werden soll.

Skizziert sieht das so aus:

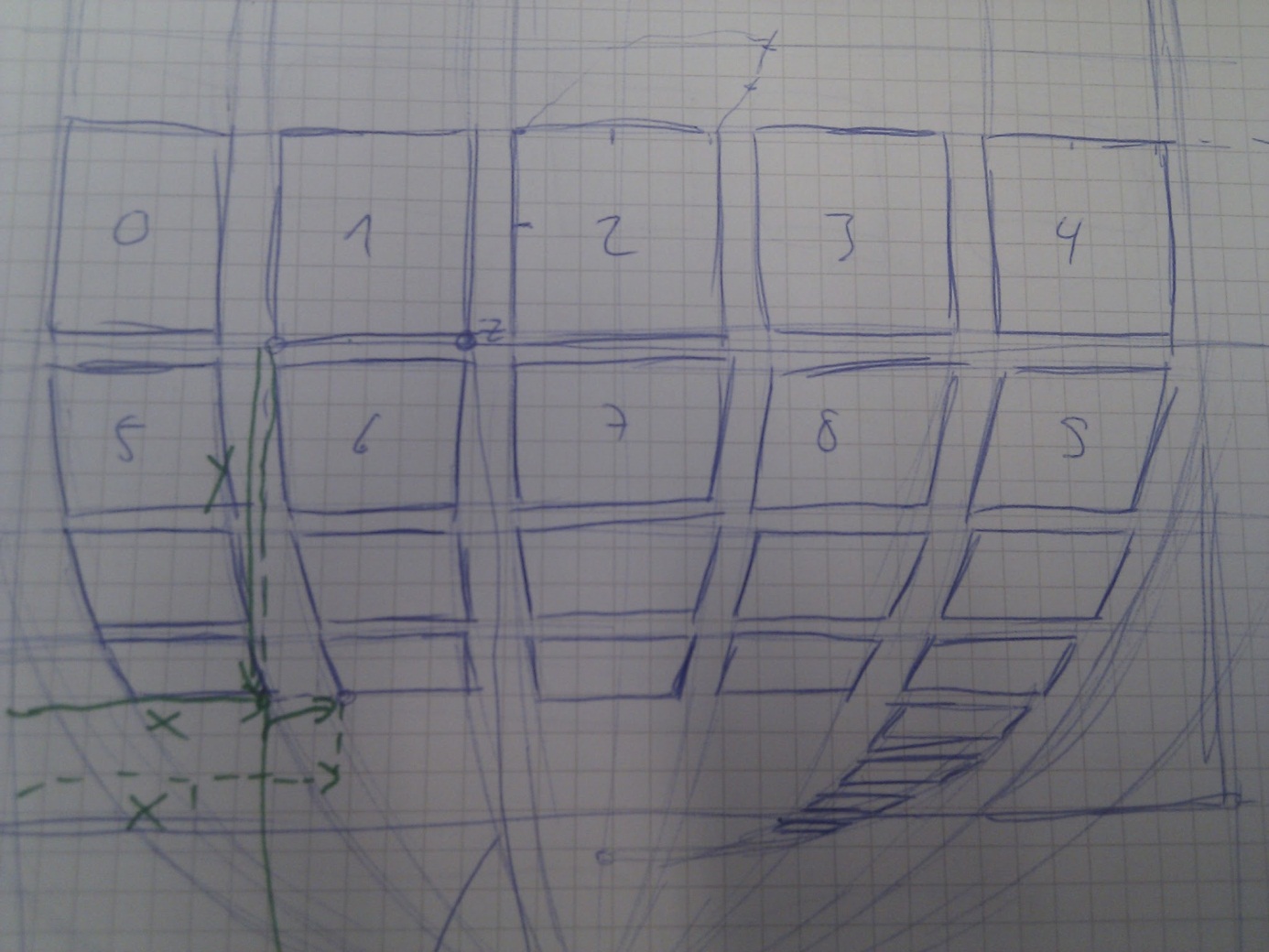


Abbildung 4 - Skizze 3D mit X, Y

Als Ergebnis wird eine Position X‘ und Y‘ zurückgegeben, also die transformierten Positionen der Project Note. Die Performance der Berechnung ist wichtig, da beim Scrolling die Berechnung jeweils für jedes Element neu durchgeführt werden muss, um die Elemente am richtigen Ort in der richtigen Grösse anzuzeigen. So muss speziell auf das Laufzeitverhalten geachtet werden, also am besten soll die Berechnung pro Element mit O(1) erfolgen. Dies ist mithilfe von Folgen und Reihen möglich:

Wobei

## Programmierung

### Implementation

Die Implementation zur Theorie stellt ein weiteres eher schwieriges Problem dar. Da die Anforderungen klar sind, die Logik aber schwierig zu implementieren war, wurde für die Entwicklung TDD (Test Driven Development) gewählt. Diese hat sich speziell bewährt, als die Implementation im Hintergrund ausgewechselt wurde, aber auch als die Performance optimiert werden musste.

Die Logik wurde in verschiedene Komponenten aufgeteilt:

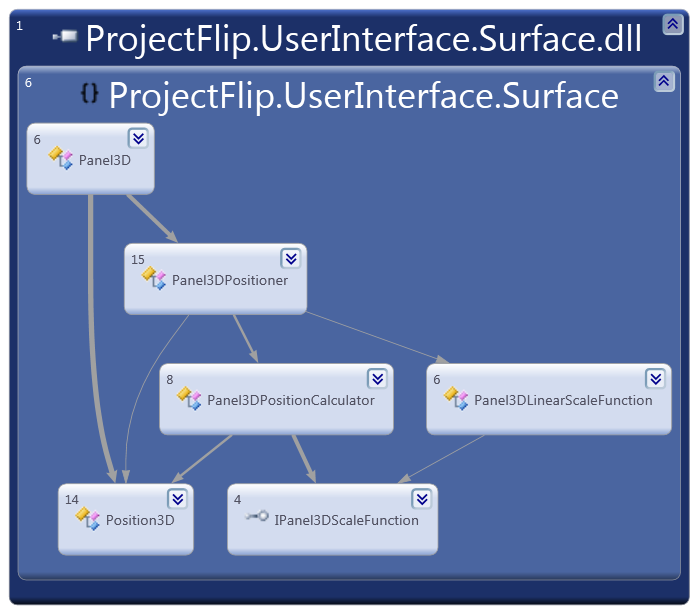


Abbildung 5 - Dependency Diagram 3D Komponenten

**Position3D**

Ein Value Object, das eine 3DPosition speichert.

**IPanel3DScaleFunction**

Stellt das Interface zur Verfügung, die eine ScaleFunction bieten muss.

**Panel3DLinearScaleFunction**

Ein **IPanel3DScaleFunction**, die linear skaliert.

**Panel3DPositionCalculator**

Der Calculator rechnet eine bestimmte Position aus, wobei er mit einer bestimmten **IPanel3DScaleFunction** arbeitet. Als Input erhält er für jede Berechnung eine bestimmte Reihe und Spalte.

**Panel3DPositioner**

Der Positioner ist dafür verantwortlich, die einzelnen Elemente im Panel anzuordnen.

**Panel3D**

Diese Komponente stellt das Panel dar. Es ist für die Events und für den Zustand des Panels verantwortlich.

### Resultat

Schlussendlich wurde in relativ wenig Zeit eine komplexe 3D Ansicht programmiert, die sowohl skaliert (mit beliebig vielen Reihen/Spalten/Fenstergrössen) wie auch performant ist. Die einzelnen Elemente können auch selektiert werden und auch die Effekte von WPF (z.B. Schatten, Rahmen) können ganz normal auf die Elemente angewendet werden.

Hier ein Beispiel des Schlussresultates:



Abbildung 6 - Vollbild Ansicht

# Weiterentwicklung

Da im Rahmen dieser Arbeit der Fokus jedoch auf anderen Aufgaben lag, konnte die Scrolling Funktionalität leider nicht implementiert werden. Deshalb musste die 3D Ansicht aufgrund Zeitmangels leider abgebrochen werden.

Falls das Scrolling noch implementiert wird, ist darauf zu achten, dass die Scroll Auswirkung im unteren Bereich vom Panel stärker ist als im oberen Bereicht. Das ist deshalb so, weil der Finger oder die Maus immer über dem gleichen Element bleiben soll wenn gescrollt wird.

Zusätzlich könnte für viele Elemente auch eine Virtualisierung noch wichtig sein, da so die Performance gesteigert werden könnte.

Weiter wäre es auch möglich, die Berechnungen zu parallelisieren, da die einzelnen Positionen ja unabhängig voneinander ausgerechnet werden können. Dies wird speziell in der Zukunft, wenn noch mehr Kerne für die Berechnung zur Verfügung stehen, wichtig sein.

# Fazit

Die Entwicklung eines eigenen Panels ist etwas aufwändig und komplex. Wenn dazu auch noch etwas in 3D dargestellt werden muss, und viel Mathematik benötigt wird, so wird es umso schwieriger. Das Ganze dann auch noch skalierbar und mit guter Performance zu implementieren, ist ein weiterer Schritt in Richtung Komplexität. Und weiter soll der Code auch noch einfach wartbar sein.

Schlussendlich muss gesagt werden, dass die Entwicklung in diesem Bereich zwar sehr anspruchsvoll war, aber auch viel Spass gemacht hat. Speziell hervorzuheben ist der Einsatz von TDD, der die Entwicklung sehr stark beschleunigt hat und durch den auch viele Fehler schnell identifiziert werden konnten.

Es ist natürlich schade, wenn eine Entwicklung nach viel Einsatz pausiert oder abgebrochen werden muss, aber in dieser Situation mit Zeitmangel war das die einzige gangbare Lösung.