|  |
| --- |
| Elmer Lukas, Heidt Christina, Treichler Delia  20. Dezember 2011 |

|  |
| --- |
| Studienarbeit |
| rototype Perspective Wall |
|  |

****

# 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 |
| 13.12.2011 | 1.2 | Abbildungsverzeichnis, Korrekturen | dtreichl |
| 20.12.2011 | 1.3 | Fokus und Kontext ergänzt | lelmer |

## Inhaltsverzeichnis

[1 Dokumentinformationen 1](#_Toc311559466)

[1.1 Änderungsgeschichte 1](#_Toc311559467)

[1.2 Inhaltsverzeichnis 1](#_Toc311559468)

[1.3 Abbildungsverzeichnis 1](#_Toc311559469)

[2 Übersicht 2](#_Toc311559470)

[2.1 Problem 2](#_Toc311559471)

[2.2 Idee 2](#_Toc311559472)

[3 Ansatz 3](#_Toc311559473)

[3.1 Genereller Ansatz 3](#_Toc311559474)

[3.2 Ausrichtung 3](#_Toc311559475)

[3.3 Anzahl Elemente und Krümmungen 4](#_Toc311559476)

[3.4 Ausblenden zu kleiner Project Notes 4](#_Toc311559477)

[4 Umsetzung 5](#_Toc311559478)

[4.1 Mathematische Grundlagen 5](#_Toc311559479)

[4.2 Programmierung 6](#_Toc311559480)

[4.2.1 Implementation 6](#_Toc311559481)

[4.2.2 Resultat 7](#_Toc311559482)

[5 Weiterentwicklung 8](#_Toc311559483)

[6 Fazit 9](#_Toc311559484)

## Abbildungsverzeichnis

[Abbildung 1 - Beispiel einer Perspective Wall, [chi91] 3](#_Toc311544326)

[Abbildung 2 - Skizze 3D ohne Krümmung, 90 Grad gedreht 4](#_Toc311544327)

[Abbildung 3 - Skizze 3D mit Krümmung 4](#_Toc311544328)

[Abbildung 4 - Skizze 3D mit X, Y 5](#_Toc311544329)

[Abbildung 5 - Dependency Diagram 3D Komponenten 6](#_Toc311544330)

[Abbildung 6 - Vollbild Ansicht 7](#_Toc311544331)

# Übersicht

## Problem

In der Übersicht über die Project Notes muss erkannt werden können, dass nach oben oder unten gescrollt werden kann. Zusätzlich wäre es schön, wenn man die Anzahl Project Notes einfach erkennen und abschätzen könnte. Man möchte also den Fokus, ein paar wenige Project Notes, aber auch den Kontext, die ganze Liste, sehen und erkennen können.

## Idee

In einer 3D-Ansicht der Project Notes kann durch den 3D-Effekt erkannt werden, dass im Hintergrund (nach oben und nach unten) noch weitere Project Notes existieren. Da die Project Notes im Hintergrund kleiner werden, kann auch die Anzahl ungefähr abgeschätzt werden. Die fokussierten Project Notes werden immer am grössten darstellt.

# Ansatz

Um eine Lösung erarbeiten zu können, wurde zuerst ein sehr genereller Ansatz gesucht. Dieser wurde dann während mehreren Schritten verfeinert. Nachfolgend wird das Vorgehen aufgezeigt.

## Genereller Ansatz

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

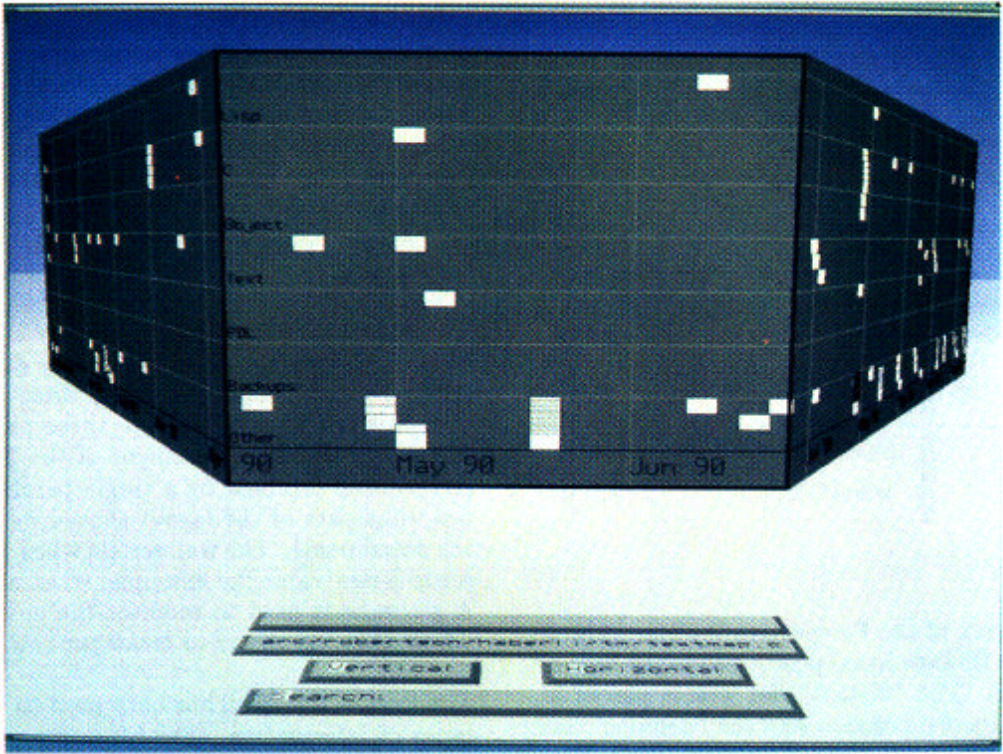


Abbildung 1 - Beispiel einer Perspective Wall, [chi91]

## 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:

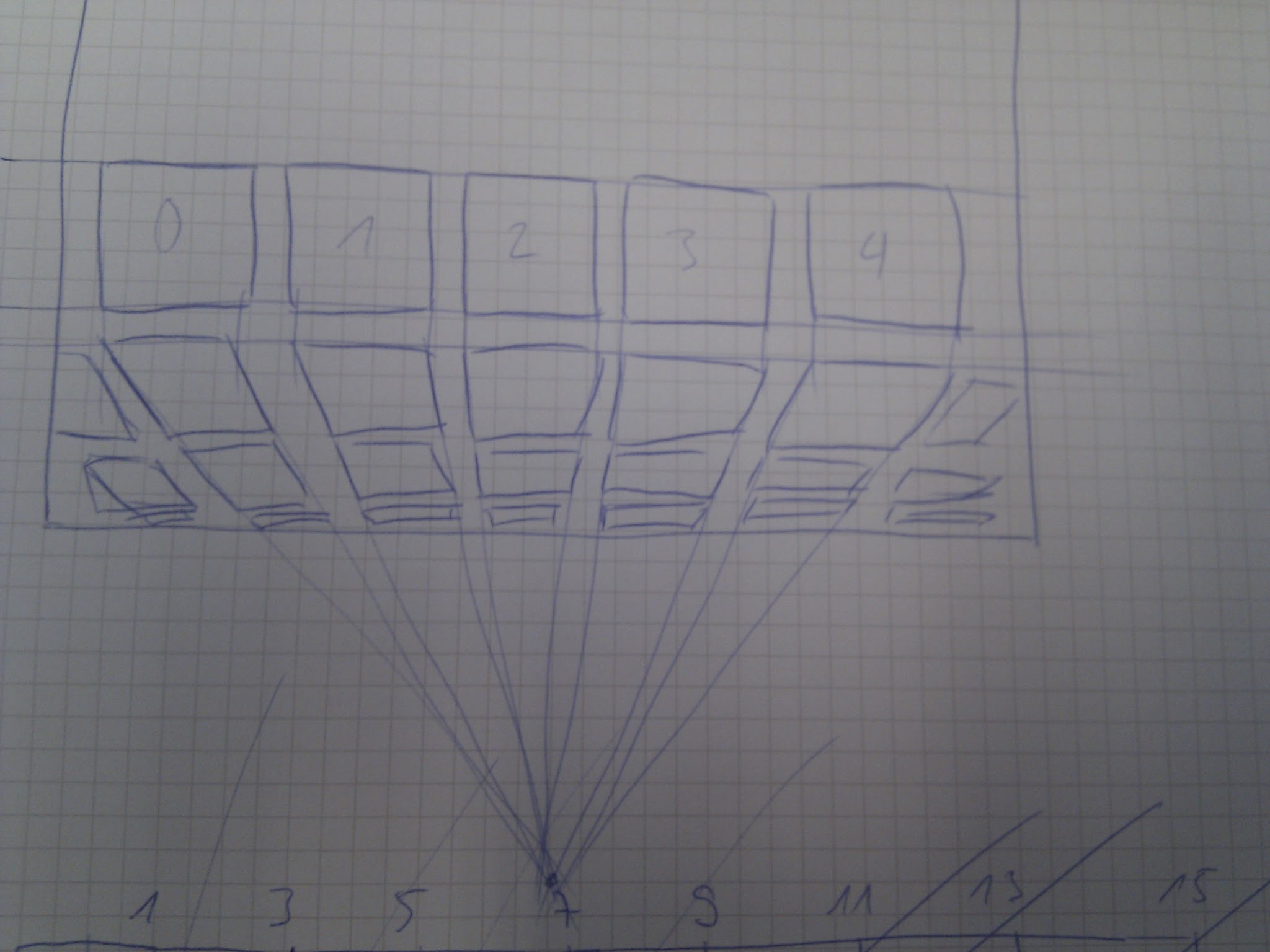


Abbildung 2 - Skizze 3D ohne Krümmung, 90 Grad gedreht

## Anzahl Elemente und Krümmungen

Die originale Perspective Wall beschränkt sich auf drei verschiedene Darstellungen: der mittlere Teil ist flach und rechteckig dargestellt, während der rechte und linke Teil der Wall perspektivisch verzogen sind.

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

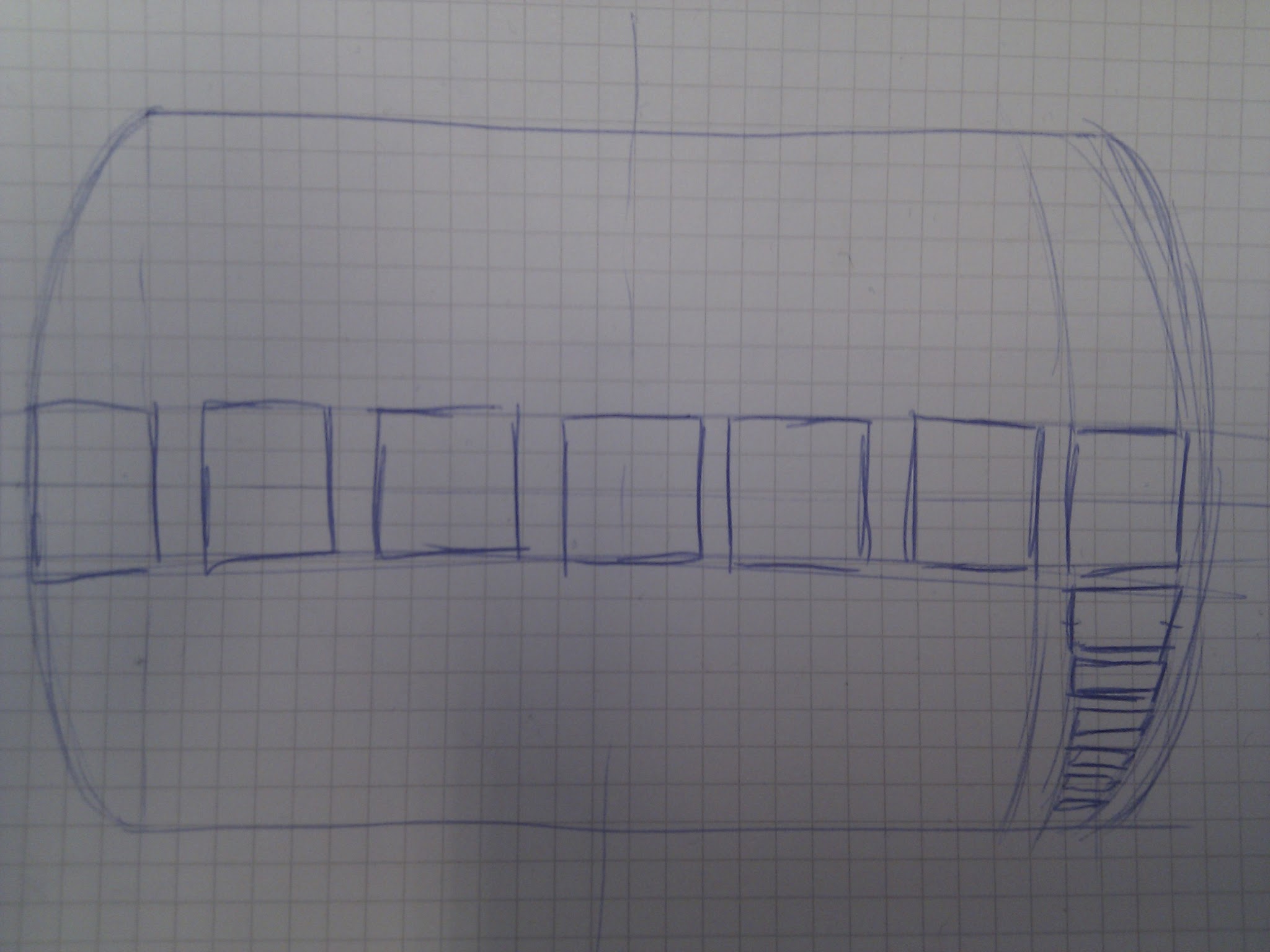


Abbildung 3 - Skizze 3D mit Krümmung

## Ausblenden zu kleiner Project Notes

Damit am unteren Rand der Darstellung keine zu stark gestauchten Project Notes vorkommen, müssen einige sich unten (aus 3D-Sicht: im Hintergrund) befindende Elemente ausgeblendet werden. Dies kann z.B. so gemacht werden, dass alle Project Notes, die weniger als 5 Pixel hoch sind, ausgeblendet werden.

Dadurch wird zwar das Prinzip von Fokus und Kontext verletzt, da Elemente ausgeblendet werden. Dies ist jedoch kein echtes Problem, da sich der Benutzer durch das 3D vorstellen kann, dass sich hinter den anderen Project Notes noch weitere, kleine Project Notes befinden könnten.

# Umsetzung

## Mathematische Grundlagen

Grundsätzlich wird eine Funktion mit zwei Parametern benötigt:

* X: die Position der X-Achse
* Y: die Höhe, auf der die Project Note gezeichnet werden soll

Skizziert sieht dies wie folgt aus:

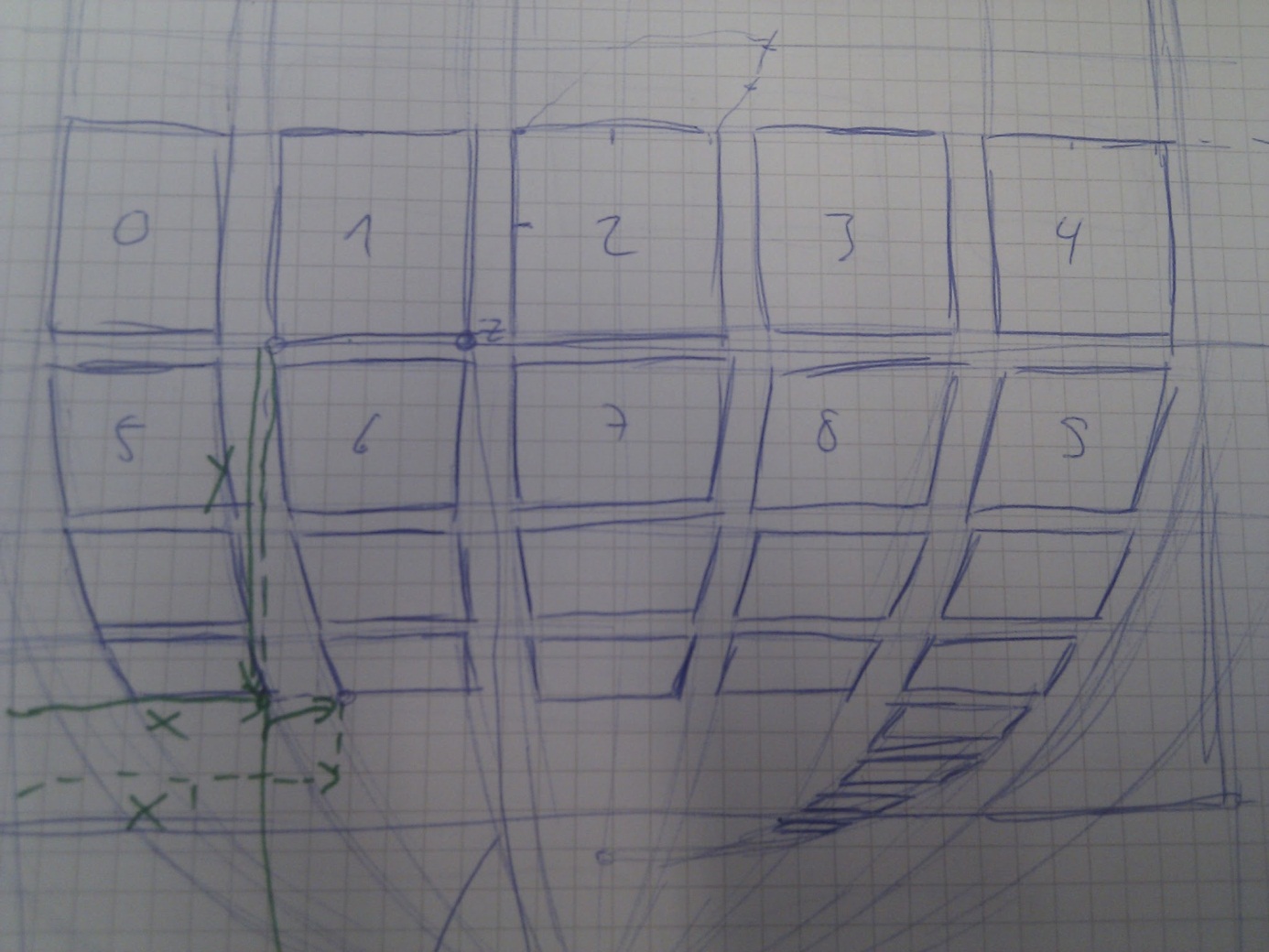


Abbildung 4 - Skizze 3D mit X, Y

Als Ergebnis wird eine Position X‘, Y‘ zurückgegeben, also die transformierte Position der jeweiligen Project Note. Die Performance der Berechnung ist wichtig, da beim Scrollen die Berechnung jeweils für jedes einzelne Element neu durchgeführt werden muss, damit die Elemente an der richtigen Position in der richtigen Grösse und Darstellung angezeigt werden. Daher muss speziell auf das Laufzeitverhalten geachtet werden, idealerweise 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 ist, wurde für die Entwicklung TDD (Test Driven Development) gewählt. Als der erste Ansatz für die Implementation der 3D-Ansicht verworfen und durch einen neuen ersetzt wurde, hat sich die TDD Methode bewährt, da die zu Beginn geschriebenen Tests weiterhin verwendet werden konnten. Auch für die Optimierung der Performance war diese Vorgehensweise hilfreich.

Die Logik wurde in verschiedene Komponenten aufgeteilt:

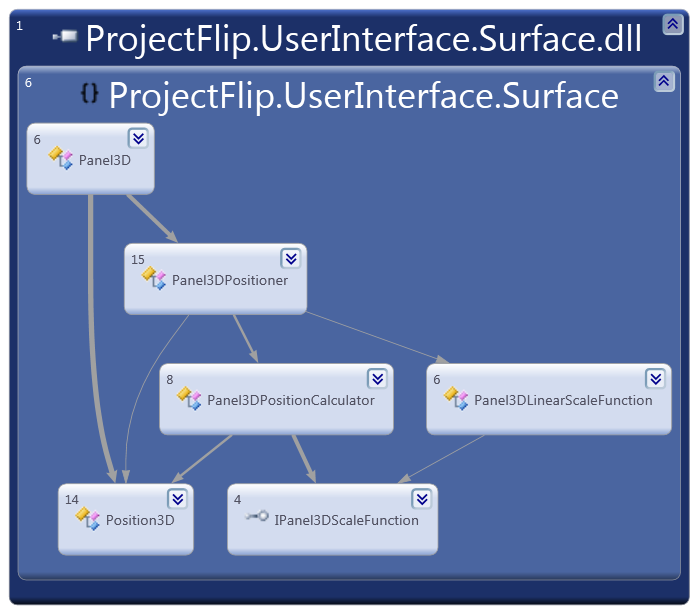


Abbildung 5 - Dependency Diagram 3D Komponenten

**Position3D**

Ein Value Object, das eine 3D-Position speichert.

**IPanel3DScaleFunction**

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

**Panel3DLinearScaleFunction**

Eine das Interface **IPanel3DScaleFunction** implementierende Klasse, 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 (X, Y Position).

**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

Das Endergebnis ist eine komplexe 3D-Ansicht, welche in relativ wenig Zeit programmiert wurde. Die Ansicht skaliert die Reihen und Spalten für beliebige Fenstergrössen und ist performant. Die einzelnen Elemente, welche sich in der Ansicht befinden, 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 der Fokus dieser Arbeit auf der Grundfunktionalität der Software lag, war die Zeit zur Entwicklung von Begeisterungsfaktoren von Beginn her begrenzt. Die Scrolling-Funktionalität der 3D-Ansicht konnte aufgrund von Zeitmangel leider nicht implementiert werden. Daher wurde das 3D-Projekt eingestellt.

Falls das Scrolling noch implementiert wird, ist darauf zu achten, dass die Scroll-Auswirkung im unteren Bereich vom Panel stärker ist als im oberen. Der Grund dafür, ist, dass sich der Finger oder die Maus während dem Scrollvorgang immer über dem gleichen Element befinden soll.

Zusätzlich könnte für viele Elemente auch eine Virtualisierung der nicht sichtbaren Elemente in der Liste wichtig sein, da so die Performance gesteigert werden kann.

Weiter wäre es auch möglich, die Positions-Berechnungen zu parallelisieren, da die einzelnen Positionen unabhängig voneinander ausgerechnet werden können. Dies wird speziell in der Zukunft, wenn 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, wird es umso schwieriger. Das Ganze dann auch noch skalierbar und mit guter Performance zu implementieren, vergrössert die Komplexität zusätzlich. Darüber hinaus soll der Code auch noch einfach wartbar sein.

Abschliessend 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 Test Driven Development (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 bzw. abgebrochen werden muss, aber in dieser Situation mit Zeitdruck für andere Funktionalitäten war das die einzige gangbare Lösung.