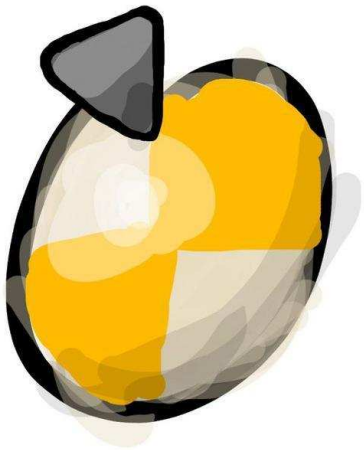
Projekt 1: TextureSync

Texturen-Sammlungen einfach und sicher verwalten

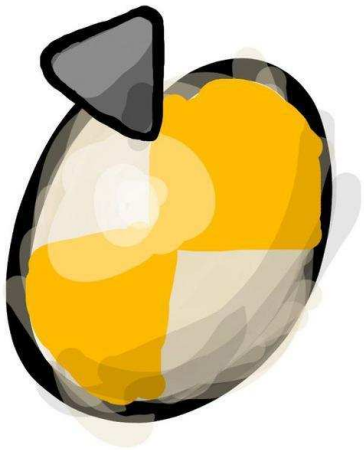


Hendrik

Einleitung:
- Begrüßung
- Unser Aufgabe des Projekt 1 war …

Agenda

- Problem
- Lösung
- Artefakte
  - Lastenheft
  - Pflichtenheft
  - Grobdesign
  - Tests
- Technologien
- Fazit
- Demo



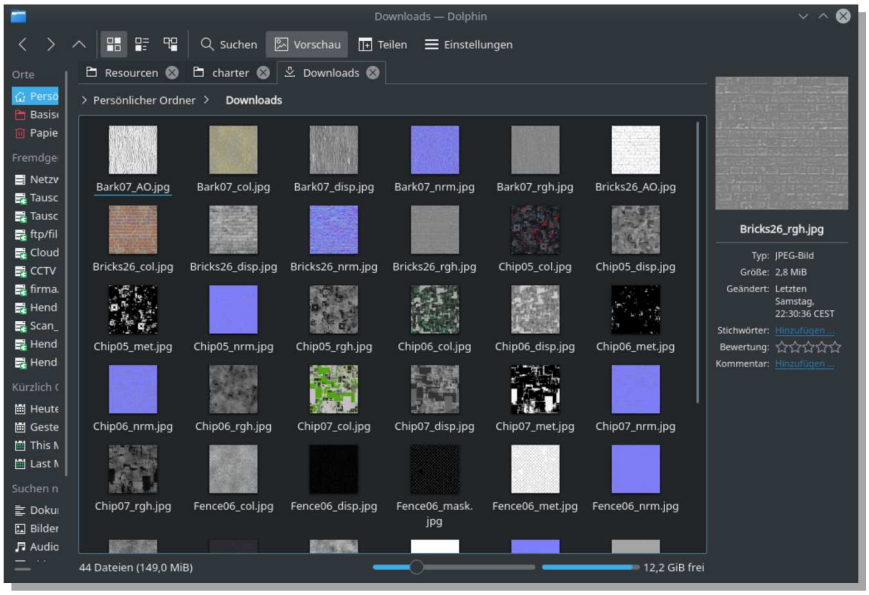
Hendrik

Auch sagen wer was macht

Problem

- Nicht durchsuchbar
- keine Sortierung
- Dezentral gespeichert
- keine Sicherung

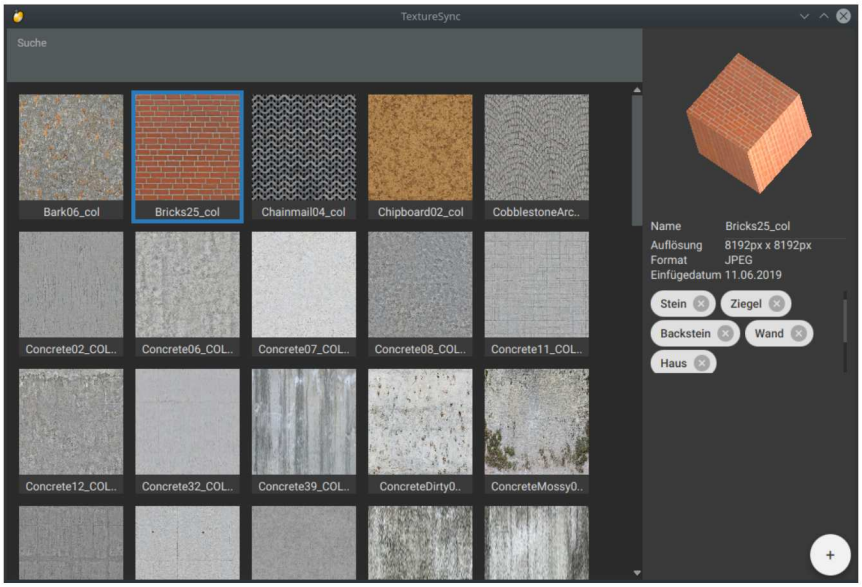
Verwaltungsaufwand



Jannik

Ausgangssituation –> schlecht

https://upload.wikimedia.org/wikipedia/commons/e/e5/Post-it-note-transparent.png



Lukas

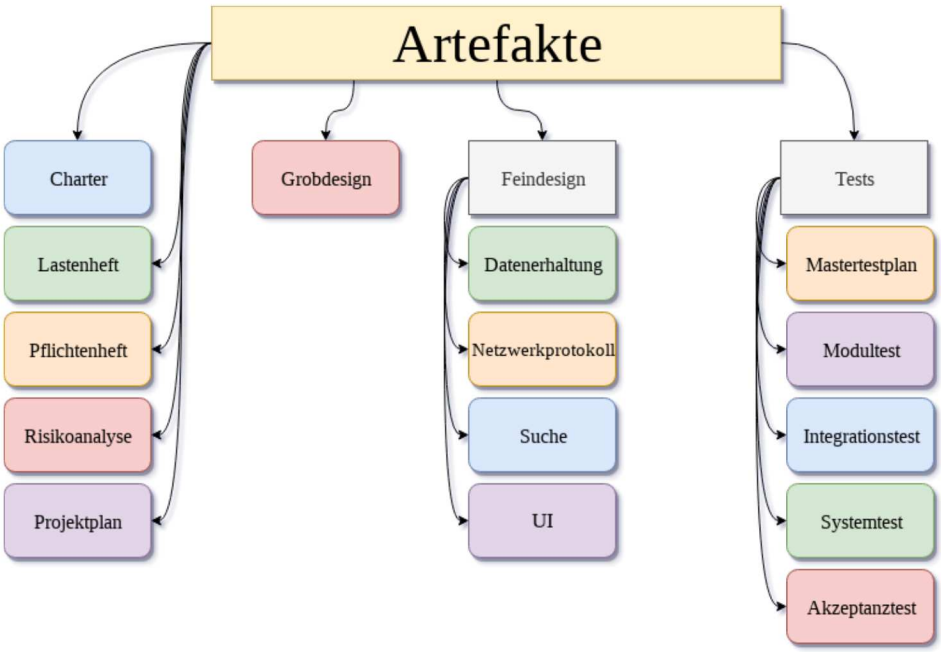
Lösung: Server-Client-Software die Verwaltung übernimmt. Speziell für Texturen optimiert (Tag-System)

Auszug aus Projekt-Charta:

Kriterien	
Wichtigste Funktionen	• Texturen anzeigen und verwalten • eine Preview beim Durchsuchen der Texturen • ein Tag-System, um Texturen zuzuordnen • Filter für Metadaten und Tags • Synchronisation mit zentralem Server
Akzeptanzkriterien	• Das Durchsuchen darf nicht länger als 1 Sekunde bei 1000 Texturen dauern. • mindestens 10 Clients gleichzeitig aktiv

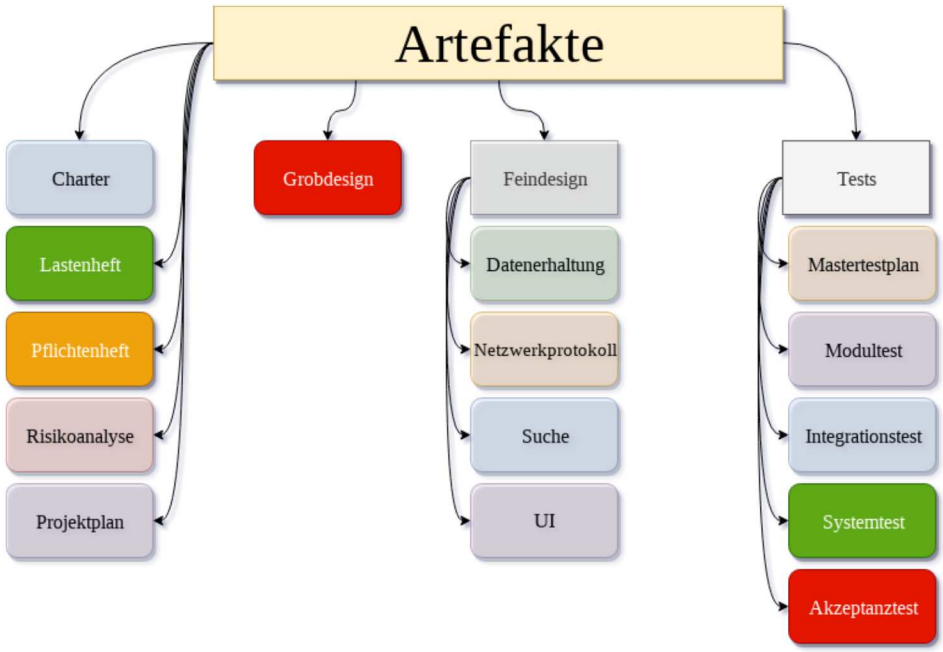
Kunden: Animationsstudios, 3D-Designer, Grafikagenturen

Robin



Robin

Alle unsere docs vorstellen und deren Hierarchie, noch kein zeigen.



Hendrik

Hervorgehobene genauer anschauen. Lastenheft und Pflichtenheft.

Jannik

Grobdesign

Lukas

System und Akzeptanztest

Alle:

Auf Traceability eingehen Anhand eines Beispiels verfolgen:

Textur importieren

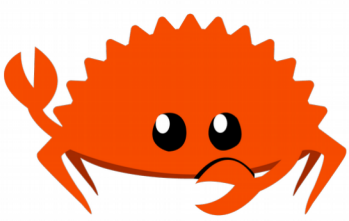
- Lastenheft F#1
- Pflichtenheft MK#7
- Grobdesign 2.2 View --> Import
- Systemtest ST#1 bis ST#6
- Akzeptanztests AT#1

Server: Rust

- Compilersprache mit ähnlicher Performance wie C++
- Compiler verhindert gängige Fehler in Programmen
  z.B. Seiteneffekte, use after free, race conditions, Pufferüberlauf
- cargo als Toolchain (ist Teil von Rust)
  übernimmt Kompilierung, Testausführung, Abhängigkeitsverwaltung

Bibliotheken:

- image: Generierung von Vorschaubildern
- serde: Speicherung der Texturdaten in Json-Datei



https://rustacean.net/assets/rustacean-flat-gesture.png

Robin

Client: Kotlin

https://logos-download.com/wp-content/uploads/2016/10/Kotlin_logo_wordmark.png

- Syntax ähnlich wie Java
- Sprachfeatures die Java nicht bietet (when mit expressions)
- Übersetzt in JVM Bytecode → läuft überall
- Native code (LLVM)
- Intellij IDEA (IDE), gradle (build tool)

Bibliotheken:

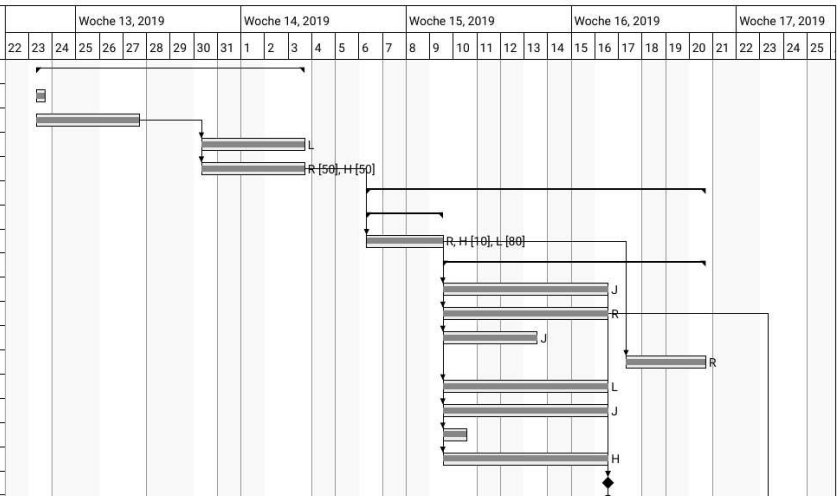
- GUI Framework: TornadoFX (JavaFX für Kotlin)
- Design lässt sich durch css ändern
- Jfoenix: Material Design GUI Elemente (ChipView, etc.)

Jannik

Pläne sind nichts, Planung ist alles.

Dwight D. Eisenhower, ehemaliger US-Präsident

Name	Arbeit
Definition	**10 T**
Charta [M]	1 T
Lastenheft [M]	3 T
Pflichtenheft [M]	3 T
Risikoanalyse [M]	3 T
Plannung	**23 T**
Grobdesign [M]	**3 T**
Architektur festlegen	3 T
Feindesign [M]	**16 T**
3D Test auf Kotlin + ChipView Test	3 T
Netzwerk-Protokoll definieren	3 T
Mockups	2 T
Wie Metadaten filtern?	2 T
Datenerhaltung	3 T
UI Elemente beschreiben	3 T
Task zuweisen	1 T
Mastertestplan [M]	3 T
End of Design	

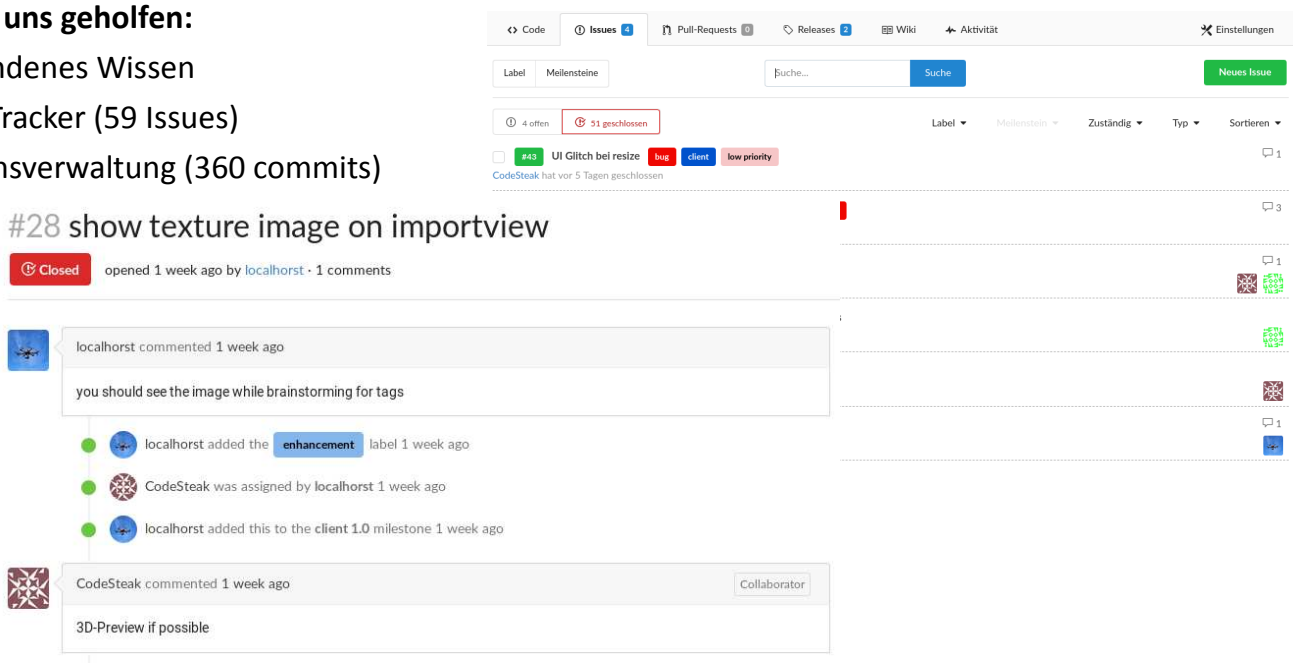


Hendrik

Fazit

Was hat uns geholfen:

- Vorhandenes Wissen
- Issue-Tracker (59 Issues)
- Versionsverwaltung (360 commits)
- Test



Hendrik

DEMO

Hendrik

Demo ankündigen

Jannik

Server und Client starten. Was zum /data Ordner des Servers sagen.

UI Elemente erklähren, aber nirgends draufklicken.

Lukas

Import einer Textur und anschließend diese Textur mittels Tag und Name suchen.

Robin

DetailView erklären und manipulieren. Export und Delete. Bedanken und nach Fragen fragen.