# Pflichtenheft

## TextureSync

| | |
|---|---|
| Version | 1.0 |
| Datum | 22.03.19 |
| Autor | Lukas Fürderer, Jannik Seiler |
| Projektmitglieder | Hendrik Schutter,<br>Lukas Fürderer,<br>Robin Willmann,<br>Jannik Seiler |

# Inhaltsverzeichnis

1 Einleitung....3
1.1 Musskriterien....3
1.2 Wunschkriterien....4
1.3 Abgrenzungskriterien....5
2 Produktumgebung....5
2.1 Architektur....5
2.2 Server....5
2.3 Client....6
3 Technologien....6
3.1 Server....6
3.2 Client....6
4 Gegenmaßnahmen für Risiken....7
4.1 Gegenmaßnahmen für Projektrisiken....7
4.2 Gegenmaßnahmen für Produktrisiken....7

# 1 Einleitung

TextureSync ist eine Client-Server Applikation zur Verwaltung und Verteilung von 3D-Texturen in einem Unternehmensnetzwerk.

Ein 3D-Designer soll hierbei so wenig Aufwand wie möglich mit der Organisation und Verteilung seiner Texturen haben. TextureSync übernimmt die zentrale Speicherung und macht erstellte Werke allen teilnehmenden Mitarbeitern des Unternehmens zugänglich.

## 1.1 Musskriterien

### Texturensammlung

Der Server kann mindestens 1000 Texturen (Grafikdateien) speichern. Jeder verbundene Client kann Grafiken hinzufügen und die Sammlung durchsuchen.

### Bildformate

TextureSync kann mindestens die Bildformate JPEG und PNG korrekt verwalten und darstellen.

### Metadaten

Beim Importieren von Texturen erfasst TextureSync automatsich die Auflösung der Textur und das aktuelle Datum als Einpflegedatum. Der Nutzer kann den Texturen bei Bedarf Tags hinzufügen und den eindeutigen Namen editieren.

### Tags

Tags sind Zeichenketten, mit denen Texturen in bestimmte Gruppen eingeteilt werden können.

Als Tags sind maximal 32 Zeichen und mindestens 1 Zeichen aus Buchstaben, Zahlen, Bindestrichen, Unterstrichen und Umlauten erlaubt. Die Groß- und Kleinschreibung (auch der Umlaute) wird nicht berücksichtigt.

Jeder Textur können beliebig viele Tags zugewiesen werden und umgekehrt kann ein Tag beliebig vielen Texturen zugewiesen werden.

Die Zuordnung der gesammelten Texturen zu ihren Tags lässt sich jederzeit von beliebigen Nutzern wieder ändern durch Hinzufügen und Löschen von Tags bei den betreffenden Texturen.

### Filter

Der Nutzer kann die am Client angezeigten Texturen nach verschiedenen Kriterien filtern. Mögliche Kriterien sind:

- Das Vorhandensein von Tags
- Das nicht-Vorhandensein von Tags
- Mindestauflösung
- Maximalauflösung
- Stichworte, die im Namen der Textur vorhanden sind

Beliebige Kombinationen dieser Kriterien sind möglich, um nur die Texturen anzuzeigen, die alle eingestellten Kriterien erfüllen.

Als Auflösung wird die Breite bzw. Höhe der Textur in Pixeln gezählt. Sollte eine Textur nicht quadratisch sein, zählt das quadratische Mittel aus Breite und Höhe.

Alle gefundenen Ergebnisse werden jeweils mit einer 2D-Vorschau aufgelistet.

### Synchronisation

Hat ein Nutzer eine Textur erfolgreich zur Sammlung hinzugefügt oder Tags zu einer Textur hinzugefügt bzw. entfernt, ist diese geänderte Textur sowie deren Metadaten für alle anderen Nutzer sichtbar.

### Export

Jede gewünschte Textur lässt sich aus der Sammlung exportieren und im lokalen Dateisystem des Clients an einem beliebigen Ort abspeichern.

### Atomarer Upload

Eine neue Textur wird erst dann in die Sammlung übernommen, wenn der Upload auf den Server vollständig und erfolgreich war. Sollte ein Upload durch einen Netzwerkausfall abbrechen, wird keine defekte Textur in die Sammlung aufgenommen.

## 1.2 Wunschkriterien

### 3D-Ansicht

Der Nutzer könnte bei Auswahl einer Textur eine 3D-Vorschau angezeigt bekommen um eine räumliche Vorstellung davon zu erhalten.

### Einfache Installation

Wünschenswert wäre, sowohl Client als auch Server, jeweils über eine einzelnes Shell-Skript installieren zu können. Dieses würde  auch die Inbetriebnahme des Server übernehmen. Alternativ könnte dies auch von einem *debian* Paket erledigt werden.

### Update durch die Paketverwaltung

TextureSync könnte als Paket in apt installiert werden, sodass es beim üblichen Systemupdate von Ubuntu ebenfalls aktualisiert wird.

### Backup

Von der Texturensammlung lässt sich im laufenden Betrieb ein konsistentes Backup erstellen, indem man ein bestimmtes, im Handbuch angegebenes Verzeichnis mit einem einfachen Kopierprogramm auf das Backuplaufwerk kopiert.

Zur Wiederherstellung des Backups reicht es aus, den Server zu stoppen, das Verzeichnis aus dem Backup zurück zu spielen und den Server erneut zu starten.

### Automatische Konfiguration des Clients

Der Client findet automatisch den Server innerhalb von 30 Sekunden, ohne dass eine IP-Adresse von Hand eingeben werden muss. Änderungen der Server IP-Adresse werden automatisch erkannt.

### 3D Ansicht als Standard Ansicht

Der Nutzer könnte ohne  Auswahl einer Textur alle gefunden Texturen als 3D-Vorschau angezeigt bekommen um eine räumliche Vorstellung davon zu erhalten..

## 1.3 Abgrenzungskriterien

### Texturen erstellen oder bearbeiten

TextureSync ist keine Bildbearbeitung. Um Texturen zu ändern, müssen diese exportiert, mit einem dafür vorgesehenen Programm bearbeitet und schließlich neu importiert werden.

# 2 Produktumgebung

## 2.1 Architektur

TextureSync ist eine Client-Server Applikation. Ein Client reicht zur Verwendung nicht aus.

Um TextureSync lokal zu verwenden, muss ein lokaler Server installiert werden, auf den der Client anschließend zugreifen kann.

## 2.2 Server

Der Server lässt sich auf einem Ubuntu Server 18.04 mit Internetzugang installieren, sofern ein *root*-Zugriff vorhanden ist.

## 2.3 Client

Der Client lässt sich auf einem Ubuntu 18.04 System mit Internetzugang und *root*-Zugriff installieren.

# 3 Technologien

## 3.1 Server

Die Server-Software wird in der Programmiersprache Rust entwickelt von Mozilla Research verwendet. Gründe hier für sind /* *insert Robin* */

Für die generierung der JSON Schnittstelle wird serde_json verwendet.

## 3.2 Client

Die Client-Software wird mit der Programmiersprache Kotlin entwickelt. Gründe hierfür ist das diese für die Java-Virtual-Machine übersetzt werden kann und so auf mehreren Plattformen lauffähig ist. Als Framework für das User Interface wird TornadoFx verwendet. Hiermit lassen sich viele moderne UI-Elemente realisieren.

# 4 Gegenmaßnahmen für Risiken

## 4.1 Gegenmaßnahmen für Projektrisiken

| | |
|---|---|
| **PjR#1** | Puffer einplanen, Worst-Case beachten. |
| **PjR#2** | Frühzeitig Prototyp bauen, testen. Anforderungen analysieren. |
| **PjR#3** | Puffer einplanen. |
| **PjR#4** | Frühzeitig Proof-of-Concept/Prototypen erstellen. Alternativen:<br>• Server erstellt Preview<br>• Wechsel des UI-Frameworks.<br>• Nur 2D-Preview |
| **PjR#5** | Aktuelle Versionen verwenden, freie Frameworks und Tools |
| **PjR#6** | Tests bei Planung definieren, während Implementieren testen |

## 4.2 Gegenmaßnahmen für Produktrisiken

| | |
|---|---|
| **PdR#1** | Möglichkeiten zum Backup schaffen; Server-Dateistruktur einfach halten. |
| **PdR#2** | Server: Nach solchen Daten beim Start scannen.<br>Client: Cache leeren beim Start oder beim Beenden. |
| **PdR#3** | Edge-Cases testen. Server sendet Nachricht, um zu zeigen, dass die Transaktion erfolgreich war. |
| **PdR#4** | Schon ausgewählte Texturen könnten lokal gecached werden. |
| **PdR#5** | Auflösung verringern, Animationen deaktivieren, nur 2D Previews |
| **PdR#6** | Bestätigung für löschen erforderlich.<br>Backup kann wieder eingespielt werden |
| **PdR#7** | Atomarer Upload, Möglichkeit Metadaten zu ändern. |
| **PdR#8** | Bestätigungs-Nachricht nach Upload an Client, nach Erfolg. User Interface informiert Nutzer über den Erfolg. |