Final Project Proposal

Duc Phan

dphan2@pdx.edu

Github URL: https://github.com/ptmdmusique/rust-ooxml-manipulator

Project: `rust-ooxml-manipulator`

The project topic area: Word and all of its shenanigans

The specific project vision. Be detailed.

A program that extracts a Word file into its OOXML representation and also provides some utils.

Read more about OOXML: Office Open XML - Wikipedia

The major features of this program include

- extract a Word file into a new folder containing its OOXML representation
- summarize the structure: how many files, how many images, how many custom XML
- re-zip into the original Word file after the user has modified the file
- allow adding/editing custom XMLs via a primitive GUI

Stretched goals: might not be included to ensure I meet the deadline

- watch for the OOXML changes to update the actual Word file live
- validate OOXML structure

The extraction is inspired by this extension OOXML Viewer - Visual Studio Marketplace

Motivation

I work with Word files daily, where we integrate content from various sources into Word documents. One of the pain points is the lack of tools to explore and edit the internal structure of those files to understand the output better, as well as fixing bugs that are not visible through the Word UI.

So this project is going to be the starting point for the tools that will be used in my work.

Feature explanation

1. Word file extraction and rezip

A Word file is a zipped representation of different file types behind the scenes. It includes

- the main OOXML in `document.xml`, representing the actual UI shown
- relationship files indicating the structural relationship between each UI element in Word
- images
- custom metadata (`customXML`) made by the users (most likely coders) programmatically (found via `item1.xml`, `itemProps1.xml`, `item2.xml`, etc)

This feature helps unzip the Word file into its mentioned internal structure and also re-zip it

2. Summarize the structure

This will analyze and summarize the file info, including

- basic file info: name, size, number of entries, other metadata
- image count, size of each, etc
- number of custom XMLs

Note that this will extract the Word file if it's not found. The result will be stored in `summary.json`

3. Analyze customXML

Analyze the `item*.xml` files to get the list of custom XML embedded in the Word file. The result will be stored in `customXML.json`

Discussion of any issues of concern you might have.

There are 2 main concerns with this project

1. The domain knowledge required to understand the business logic

In order to make the process of grading easier, I'll include the test files with specific instructions to help with testing. I'll also include explanations through documentation inside each function itself. Moreover, I'll do my best to also architect the program to be as modular as possible to make each utility intuitive to understand

2. The scope of the actual implementation

To complete all the proposed features, including the stretch goal, 3 weeks is nowhere near enough. Therefore, I'll focus on the primary goals and simplify the implementation as much as possible while ensuring stability.