Rapport: ASCII Art Studio

1. Implementerad uppgift

Jag har utvecklat ett ASCII Art Studio - ett program som konverterar vanliga bildfiler till ASCII-konst. Programmet ger användaren möjlighet att ladda bilder, konvertera dem till ASCII-representationer med anpassningsbara tecken, samt visa information om de laddade bilderna. Lösningen inkluderar ett interaktivt kommandobaserat gränssnitt som gör det lätt att använda och experimentera med olika teckenkombinationer för att skapa unika ASCII-konstverk.

2. Programstart och användning

Programmet startas genom att köra huvudfilen med Python:
python3 ascii_art_studio.py
Efter start presenteras användaren med en kommandorad där följande kommandon är tillgängliga:

- load <filnamn> - Laddar en bildfil (t.ex. load bild.jpg)
- render - Renderar den laddade bilden som ASCII-konst
- info - Visar information om den aktuella bilden
- set_chars <tecken> - Ställer in anpassade tecken för ASCII-konvertering (t.ex. set_chars @%#*+=-:.)
- quit - Avslutar programmet

Exempel på en komplett användningssession:
AAS: load landscape.jpg
Image 'landscape.jpg' loaded.
AAS: set_chars @%#*+=-:.
ASCII characters set to: '@%#*+=-:.'
AAS: render
[ASCII-konsten visas här]
AAS: info
Filename: landscape.jpg
Size: (800, 600)
Current ASCII characters: '@%#*+=-:.'
AAS: quit
Bye!

3. Använda bibliotek och installation

Programmet använder följande externa bibliotek:

- PIL (Pillow) - För bildbehandling och manipulation

Pillow är inte en del av Pythons standarddistribution och måste installeras separat med pip:
pip install Pillow
Övriga använda moduler (som sys och os) ingår i Pythons standardbibliotek och kräver ingen extra installation.

4. Programstruktur och arkitektur

Programmet är strukturerat i en enda fil (ascii_art_studio.py) med en huvudsaklig klass:

ASCII Art Studio-klassen

Huvudklassen som hanterar hela program logiken och tillståndshantering:

- __init__(self) - Initierar programmet med tom bild, filnamn och standard-ASCII-tecken
- run(self) - Huvudloop som läser och hanterar användarens kommandon
- load_image(self, filename) - Laddar och konverterar bildfiler till gråskala
- set_chars(self, chars) - Möjliggör anpassning av tecken för ASCII-konvertering
- render_image(self) - Konverterar bilden till ASCII-konst och skriver ut resultatet
- print_info(self) - Visar information om den laddade bilden och aktuella inställningar

Programflöde

1. Programmet initieras och skapar en instans av ASCII Art Studio-klassen
2. Huvudloopen startar och presenterar användaren med en kommandorad
3. Användaren matar in kommandon som parsas och exekveras
4. Vid bildladdning konverteras bilden till gråskala och sparas i minnet
5. Vid renderingskommando skalar programmet om bilden och konverterar varje pixel till ett ASCII-tecken
6. Informationen visas i konsolen i realtid

Reflektioner

Kod Design och arkitekturval

Jag valde en objektorienterad design med en enda klass som encapsulerar all funktionalitet. Detta gör koden lätt att underhålla och utvidga med nya funktioner. Användargränssnittet är enkelt och textbaserat, vilket gör programmet lätt att använda utan krångliga grafiska gränssnitt. Valet av att separera olika funktioner i metoder med tydliga ansvarsområden gör koden mer läsbar och testbar.

Algoritmer och datastrukturer

Programmet använder flera algoritmer för att konvertera bilder till ASCII-konst:

1. Bildladdning och konvertering - Använder Pillow-bibliotekets optimerade funktioner för att läsa bildfiler och konvertera dem till gråskala
2. Bildskalning - Skalar om bilder till en fast bredd (50 pixlar) med bevarade proportioner genom att använda höjd-/bredd förhållandet
3. Pixel-till-tecken-mappning - Mappar varje pixels ljusstyrka (0-255) till ett tecken i en användardefinierad teckenuppsättning med linjär interpolering

Datastrukturer:

- Listor - För att lagra ASCII-konsten rad för rad
- Strängar - För att representera teckenuppsättningar och enskilda rader i ASCII-konsten
- Tupler - För att hantera bild dimensioner (bredd, höjd)

Prestanda Överväganden

- Tidseffektivitet: Bildbehandling Operationerna har O(n) komplexitet där n är antalet pixlar i den skalade bilden
- Minnes Effektivitet: Programmet laddar endast en bild i taget i minnet och skalar ner den till en hanterbar storlek (50 pixlar bred) innan konvertering, vilket minimerar minnesanvändningen

Användaranpassning: Möjligheten att ange egna tecken ger stor flexibilitet utan att påverka prestanda

Programmet är optimerat för att hantera bilder effektivt genom att skala ner dem till en lämplig storlek innan konvertering till ASCII-konst. Användaren kan enkelt experimentera med olika teckenkombinationer för att uppnå olika visuella effekter utan att behöva ändra koden.