Text-to-Art Interpreter: System Requirements Specification

By Lawrence Dickey, Wallis Muraca, Adam Carlson

Revision History:

Date	Version	Description	Authors
10/08/17	1.0	Initial version of document	Lawrence Dickey, Wallis Muraca, Adam Carlson
11/17/17	2.0	Rewritten document according to feedback	Lawrence Dickey, Wallis Muraca, Adam Carlson

Version 2.0

Table of Contents:

List of Figures:

List of Tables:

1. Introduction

1.1 System purpose:

We seek to create software which allows users to transform literary writings into new artistic forms such as drawings or music. Especially in the artist community, there is a desire to look at already existing works of art, such as writings, and convert them into something new. These transformations allow both artists and others to gain a new perspective on the original writing. In addition, the software can be tweaked to an artist's liking in order to better align with their style when transforming such a writing.

1.2 System scope

In this project, we hope to getting the first running version of our software, described at a high level above, working. This version 1.0 will not have all of the customizable features someone may want. However, at its core, the interface will allow users to at least read in some form of writing and convert it into a drawing. In addition, there will be beta level implementations of converting the writing into some form of music. The graphic user interface will be basic but usable for the regular individual.

1.3 Definitions, acronyms, and abbreviations

There are no unusual definitions, acronyms, or abbreviations involved with our project.

1.4 References

We did not require the use of references for this novel idea.

1.5 System overview

We believe this version 1.0 system can be completed in 4-6 weeks. The software will be primarily written in Python in order to create the GUI and working implementation. Time permitting, the Python code will also interact with a database in order to store user information and settings.

2. General system description

2.1 System context

In general, the major features that the system should implement is allowing a user to read in a file, and generating an appropriate image.

2.2 System modes and states

The system will create drawings and music. Drawing is the most essential capability, and music will be added if there is time. Use case diagrams below describe ways in which a user might use the system.

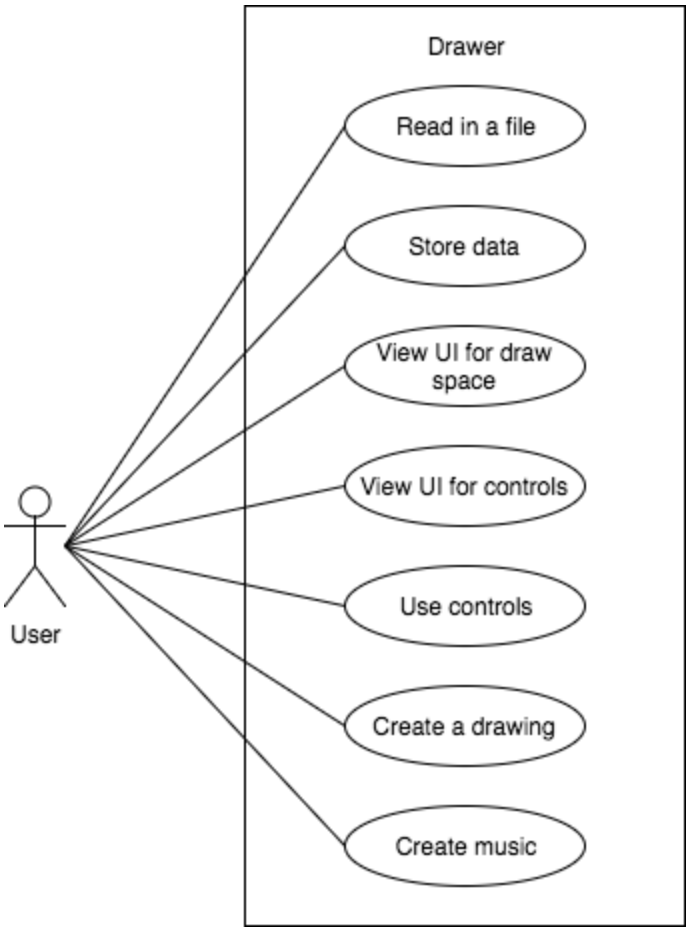

Figure 1: High level use case for entire system

These activities can occur in any order. Therefore, instead of numbering them below in our table, we created letters to represent any given action

Table 1: Use Case 1 - High level use case for entire system description
Primary actors: User **Preconditions**: User has an available text file on their computer to use **Basic flow of events:** 1. User reads in a new text file. 2. The data is stored. 3. User views draw space. 4. User views controls.

5. User may use different controls.
6. User can create a drawing.
7. User can create music.

Alternative flows:

1a. User chooses an invalid file (not a .txt).

1a1. An error message is printed saying, "Please use a text file."

6a. User has not loaded any files before trying to create a drawing.

6a1. An error message is printed, “Please upload or select a file first.”

7a. User has not loaded any file before trying to create a song.

7a1. An error message is printed, “Please upload or select a file first.”

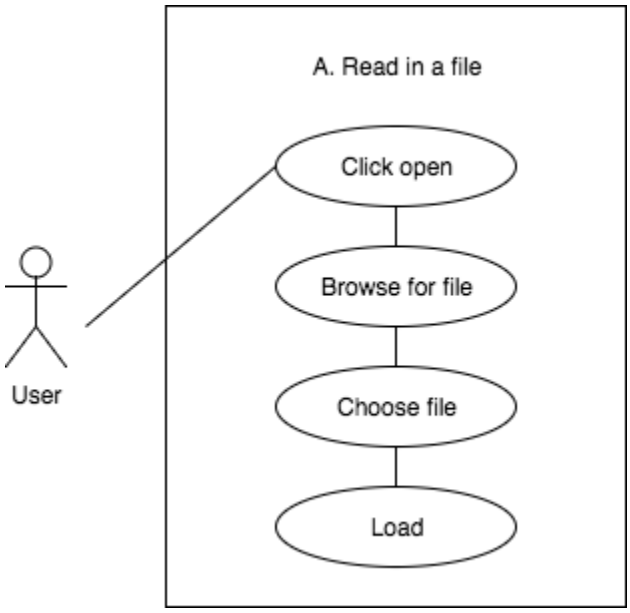

Figure 2: Low level use case for A. Reading in a file

Table 2: Use Case 2 - Low level use case for A. Reading in a file
Primary actors: User **Preconditions:** User has an available text file on their computer to use

Basic flow of events:
1. User clicks "Open New Text File."
2. An Open File dialog is displayed.
3. User chooses a file from their hard drive.
4. User clicks "Open.
5. The dialog closes.
6. The characters are parsed and stored in an external text file.

Alternative flows:
2a. User clicks "Cancel"
    2a1. Nothing happens.
4a. User chooses an invalid file (not a .txt)
    4a1. An error message is printed, "Please use a text file."
    4a2. Nothing happens, and user must click "Open New Text File."
    4b1. Dialog closes and nothing happens.

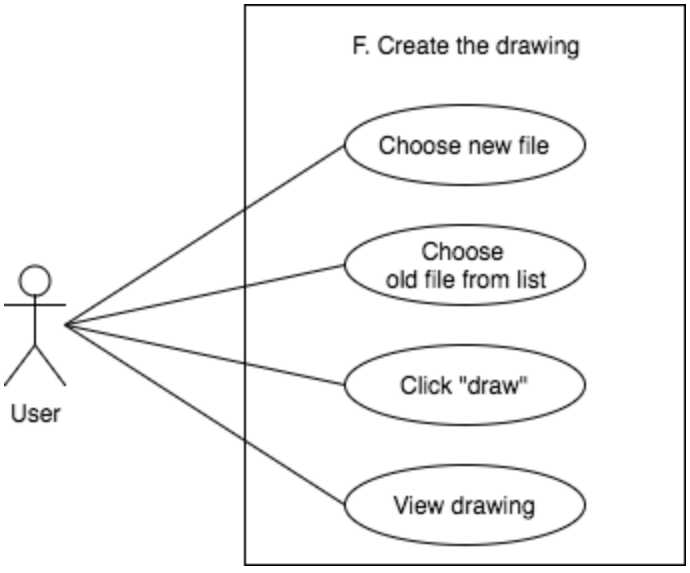

Figure 3: Low level use case for F. Create the drawing

Table 3: Use Case 3 - Low level use case for F. Create the drawing
Primary actors: User

Preconditions: User has at least one file loaded
User has selected a file

Basic flow of events:
1. User selects the text file from the list they wish to visualize.
2. User clicks "Draw Picture of Text File."
3. Drawing is rendered in the canvas.

Alternative flows:
1a. User does not select a file.
1a1. The most recently selected file is drawn.
1b. User has not loaded any files
1b1. An error message is printed, “Please upload or select a file first.”
2a. User does not click "Draw Picture of Text File."
2a1. Nothing happens.

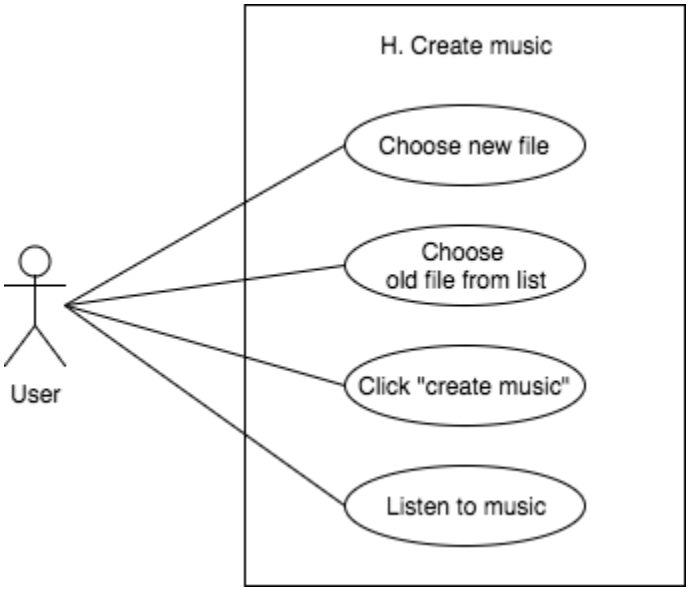

Figure 4: Low level use case for H. Create music (an optional requirement that will be implemented if there is time)

Table 4: Use Case 4 - Low level use case for H. Create music
Primary actors: User **Preconditions:** User has at least one file loaded User has selected a file **Basic flow of events:** 1. User selects the text file from the list they wish to create a song with. 2. User clicks "Create Music." 3. Music is played. **Alternative flows:** 1a. User does not select a file. 1a1. The most recently selected file is used to make a song. 2a. User does not click "Create Music." 2a1. Nothing happens.

2.3 Major system capabilities

2.3.1 Read in a file

2.3.1.1 Functional: The user shall be able to locate a file in their file browser.

2.3.1.2 Non-functional: The system shall have a reliable and easy-to-use UI to open a file.

2.3.2 Store data in a useful way

2.3.2.1 Functional: The system shall store the user's input in a useful way.

2.3.2.2 Non-functional: The system shall be able to comfortably handle up to 10,00 characters.

2.3.3 Build a UI for the draw space

2.3.3.1 Non-functional: The system shall contain a UI that includes a draw space and settings panel to modify the drawing conditions.

2.3.4 Create user controls (buttons, etc.)

2.3.4.1 Functional: The user shall be able to modify the parameters for the drawing.

2.3.4.2 Non-functional: The system shall feature a control panel for users to modify settings for the drawing

2.3.5 Create the drawing

2.3.5.1 Functional: The user shall be able to create a drawing based on their text file.

2.3.5.2 Functional: The user shall be able to save their drawing as an image file.

2.3.6 Create music

2.3.6.1 Functional: The user shall be able to create music based on their text file.

2.3.6.2 Functional: The user shall be able to pause and play the music.

2.3.6.3 Non-functional: The system shall be able to produce different notes.

2.4 Major system constraints

One major constraint is that Python can be very slow, and could struggle to handle a large number of commands or images (this is exacerbated by any kind of graphical functionality). To alleviate this, the user could be instructed to use smaller file sizes, or the system could automatically truncate after a certain number of characters.

2.5 User characteristics

There are no different interfaces or use cases for different users at the present moment.

2.6 Operational scenarios

Currently there are only two major scenarios. Both involve a user loading up the system, locating a file, and loading it into the software. Once the file is loaded, a drawing will be created. A different scenario will exist where the user may also create music. Additional extra features may also be implemented later if there is enough time, and this could create new scenarios.

3. System capabilities, conditions, and constraints

3.1 Physical

No special physical requirements exist.

3.2 Construction

The software will primarily be written in Python, and additional libraries will be used (such as Turtle and GUI libraries).

3.3 Durability

The software should not deteriorate quickly, because it does not rely on other major software or changing infrastructure. Currently, only a very small part of it relies on the Internet.

3.4 Adaptability

As text input is unlikely to change dramatically, the software should not have to adapt much.

3.5 System performance characteristics

As mentioned above, the system should be able to comfortably handle 10,000 characters and 100 frames of an animated GIF.

3.6 System security

The only minor security risk is the extra social feature, which would require using social media services' APIs.

3.7 Information management

Currently, the user would only work with one file at a time. However, it might make to sense to implement a feature where the user can navigate between different files.

4. System interfaces

The interface will consist primarily of the draw space, with a toolbar. The toolbar will provide tools for the user to change the drawing parameters. There will be a toggle to switch between music on or off. As more tools are added with extra time, there might be a “splash” page with the different options, or a menu option to switch.

Version 1.0

Software Requirements Specification - Python Drawer

1. Introduction

- System purpose
    - We seek to create software which allows users to transform literary writings into new artistic forms such as drawings or music. Especially in the artist community, there is a desire to look at already existing works of art, such as writings, and convert them into something new. These transformations allow both artists and others to gain a new perspective on the original writing. In addition, the software can be tweaked to an artist's liking in order to better align with their style when transforming such a writing.
- System scope
    - In this project, we hope to getting the first running version of our software, described at a high level above, working. This version 1.0 will not have all of the customizable features someone may want. However, at its core, the interface will allow users to at least read in some form of writing and convert it into a drawing. In addition, there will be beta level implementations of converting the writing into some form of music. The graphic user interface will be basic but usable for the regular individual.
- Definitions, acronyms, and abbreviations
    - There are no unusual definitions, acronyms, or abbreviations involved with our project.
- References
    - We did not require the use of references for this novel idea.
- System overview
    - We believe this version 1.0 system can be completed in 4-6 weeks. The software will be primarily written in Python in order to create the GUI and working implementation. Time permitting, the Python code will also interact with a database in order to store user information and settings.

2. General system description

- System context
    - In general, the major features that the system should implement is allowing a user to read in a file, and generating an appropriate image.
- System modes and states

    - The system will create drawings and music.
- Major system capabilities
    - Functional requirements:
        - Essential:
            - The user shall be able to read in a file.
            - The user shall be able to view a generated drawing.
            - The user shall be able to save the drawing as an image file.
        - Non-essential
            - The user shall be able to share the image through social media.
            - The user shall be able to save the image as an animated GIF.
            - The user shall be able to produce music through their input.
    - Non-functional requirements:
        - Essential
            - The system shall store the user's input in a useful way.
            - The system will be able to comfortably handle up to 10,000 characters.
        - Non-essential
            - The system will be able to comfortably generate animated GIFs of up to 100 frames.
- Major system constraints
    - One major constraint is that Python can be very slow, and could struggle to handle a large number of commands or images (this is exacerbated by any kind of graphical functionality). To alleviate this, the user could be instructed to use smaller file sizes, or the system could automatically truncate after a certain number of characters.
- User characteristics
    - There are no different interfaces or use cases for different users at the present moment.
- Operational scenarios
    - Currently there are only two major scenarios. Both involve a user loading up the system, locating a file, and loading it into the software. Once the file is loaded, a drawing will be created. A different scenario will exist where the user may also create music. Additional extra features may also be implemented later if there is enough time, and this could create new scenarios.

3. System capabilities, conditions, and constraints

- Physical

    - No special physical requirements exist.
- Construction
    - The software will primarily be written in Python, and additional libraries will be used (such as Turtle and GUI libraries).
- Durability
    - The software should not deteriorate quickly, because it does not rely on other major software or changing infrastructure. Currently, only a very small part of it relies on the Internet.
- Adaptability
    - As text input is unlikely to change dramatically, the software should not have to adapt much.
- System performance characteristics
    - As mentioned above, the system should be able to comfortably handle 10,000 characters and 100 frames of an animated GIF.
- System security
    - The only minor security risk is the extra social feature, which would require using social media services' APIs.
- Information management
    - Currently, the user would only work with one file at a time. However, it might make to sense to implement a feature where the user can navigate between different files.

4. System interfaces
- The interface will consist primarily of the draw space, with a toolbar. The toolbar will provide tools for the user to change the drawing parameters. There will be a toggle to switch between music on or off. As more tools are added with extra time, there might be a "splash" page with the different options, or a menu option to switch.