Product Specification: blender-hand-drawn-npr

July 19, 2018

Contents

1 Revision History

- B (this version): Customer comments incorporated. Requirement numbering frozen. Risks, User Stories and Coverage Matrix populated.
- A: First issue for Customer review and completion of Questionnaire.

2 Purpose

This document defines the Functional and Non-Functional requirements of the System. Critical statements are given unique IDs as follows:

- Functional requirements are indicated with prefix *F*.
- Non-Functional requirements are indicated with prefix *N*.
- Clarifications to initial requirements are indicated with prefix *C*.
- Assumptions are indicated with prefix *A*.

Functional requirements will form the basis of User Stories (Section 8). The Coverage Matrix (Appendix A) provides a mechanism to ensure all Functional requirements are captured as User Stories.

3 Initial Requirements

3.1 Project Brief

This project will look at augmenting 3D rendering software to produce [*N10: high-quality*] [*N20: scientific*] [*N30: 3D*] surfaces with [*N40: pseudo hand drawn*] appearance. The project will focus on extending the [*N50: Blender*] [blender.org] renderer to produce these graphs, most likely via [*N60: Python*] scripting.

Traditional hand drawn plots (see below) are able to [*N70: reveal structure*] in 3D surfaces that is often lost in modern renders. Although modern renders have accurate light transport models, they are designed for photo realism rather than to reveal the structure of surfaces. This is particularly relevant when producing figures for [*N80: reproduction*], which must be [*N90: clear*] and might only be [*N100: monochrome*].

Traditional artists developed techniques to [*N110: reveal shading*] and surface features. The aim of this project will be to develop a system to produce high quality images [*N120: automatically*], according to a specification provided by a user (e.g. [*F10: line-only*], [*F20: highlight creases*], [*F30: no lighting*]).

3.2 Customer Meetings

3.2.1 18-June-2018: Kick-Off (Week 1)

- A [*N130: blender add-on*] is to be developed which will produce images of hand-drawn appearance.
- [*N140: Animation will not be supported*], i.e. [*N150: temporal cohesion is not a concern*].
- [*N160: 3D model will be the input*].
- [*N170: Vector SVG will be the output*].
- Drawing style shall reveal surface details and shape, [*F40: regions of high-curvature*] etc.
- [*N180: A specific drawing style shall be chosen*], although [*N190: system design shall be flexible enough to add additional styles*] at a later date.
- [*N200: No requirement to reveal surface texture*].
- [*F50: Lines shall be scaled according to distance of the camera from the object*].
- The approach is assumed to require Python scripting to [*N210: process render layers produced by the blender rendering engine*].
- A nice feature to have could be to [*F60: mark areas of the model which must be rendered with a line/pattern*] (e.g. to draw attention to specific areas of interest, or to allow non-deterministic output).
- User interaction will be via the existing Blender GUI. [*N220: Custom Blender GUI panels shall be developed as required to support all functionality.*]

3.2.2 25-June-2018: Weekly (Week 2)

- Focus is on black and white images, however it may be useful to look at [*F70: limited use of colour*] (2-3 colours max).

4 Questionnaire

4.1 Non-Functional Requirements

4.1.1 Clarification of Initial Requirements

C10. **Ref N10, define the term "high-quality".**
[*F80: Parsimony of ink: draw only the strokes which are needed and not superfluous ones.*] Strokes should reveal structure.

[*F90: Accuracy of curvature: strokes should not be excessively smoothed, nor excessively rough and should reflect the underlying geometry.*]

[*F100: Consistency of lines: lines should not be unnaturally broken or incorrectly merged.*] [*F110: Strokes should continue together cleanly.*]

[*F120: Accuracy of joints: points where strokes merge (e.g. corners) should be accurately aligned, not offset messy crossings.*]

[*F130: Variation of strokes: stroke thickness should vary in a way that reveals structure, e.g. lighting or perspective foreshortening.*]

[*F140: Consistency of reproduction: small changes in mesh or view should result in small apparent changes in drawing (but temporal coherency, as for animations, is not required; randomness is quite acceptable).*]

C20. **Ref N50, state the minimum Blender version number to support.**
Blender 2.79

C30. **Ref N100, interpretation is "monochrome" means strokes will be rendered in a single colour, rather than in tones of a single colour. Please clarify if otherwise.**
Primary goal will be B/W rendition (i.e. solid black strokes on white background). Optionally, [*F150: strokes of a number of other shades may be included as an optional requirement, for example to indicate lighting of different colours on different facets.*]

C40. **Ref N120 and N160, interpretation is the the User will configure a Blender scene with a 3D surface mesh, including creation and positioning of a camera and any required lighting. This will be the starting point for interaction with the System. Please clarify if otherwise.**
User will create the Blender scene. Software will be a pass after the scene has been completely created (modelled, lit, camera view set, etc.).

C50. **Ref N170, state the minimum SVG version number to support.**
SVG 1.1.

C60. **Ref N180, development will focus on producing stroke-based illustrations in the Pen-and-Ink style, which aligns with requirements N10, N20, N30, N40, N70, N80, N90, N100, N110, F10, F20, F30 and F40. If another illustration style is thought to be better suited, please state it here.**
Stroke based rendering is the priority. Other rendering styles might include cross-hatching.

C70. **Ref N220, if there is a preference for where the GUI panels should be located within the Blender interface, please state so here.**
No strong preference. Probably as a new panel in the right hand panels, or perhaps at the bottom of the render panel?

4.1.2 Additional Requirements

N230. **Which operating systems shall be supported? Please also indicate minimum version numbers.**
Windows 10 support desired and should be tested; Ubuntu Linux as a lower priority; ideally no OS-specific requirements.

N240. Please rate the relative importance of each of the following characteristics:

- Functionality.
- Reliability.
- Usability.
- Portability.
- Efficiency.
- Maintainability.

N250. **In a few sentences, state the most critical measure of success for the System, i.e. what does a successful product look like?**
A user opens a Blender scene. They click a button and an SVG document is produced that produces an aesthetically pleasing, accurate representation of the scene, using only strokes, with a parsimony of ink and effective communication of the surface features.

N260. **It is assumed that only a single mesh will be present in the Scene to be rendered. Please clarify if otherwise.**
Single mesh is reasonable assumption.

N270. **If there are other non-functional requirements not captured by the sections above, please state them here.**
None I can think of.

4.2 Functional Requirements

4.2.1 Clarification of Initial Requirements

C80. **Ref F10, interpretation "line-only" means only the object outline/silhouette will be rendered. Please clarify if otherwise.**
Line-only could include synthesised lines e.g. to show contours which would not fall on the silhouette.

C90. **Ref F20, interpretation is "highlight creases" means only edges whose neighbouring faces meet at an angle greater than a User-defined value will be rendered. Please clarify if otherwise.**
Creases would be defined in terms of curvature (either the geometric curvature of the mesh, or the visual apparent curvature) rather than in terms of angles.

C100. **Ref F30, interpretation is "no lighting" means only geometric data (or world data such as ambient occlusion) would be used to determine the placement of feature-highlighting strokes. Please clarify if otherwise.**
Actually I think lighting is fairly important and should be included. In particular [*F160: it should be possible to adjust density or thickness of strokes according to incident light, either diffuse direct light, or to emphasise areas using AO information.*]

C110. **Ref F50, should the User have the option to toggle this stroke tapering effect on and off?**
User should be able to control stroke effects.

C120. **Ref F50, should the User have the option to influence degree of stroke taper by controlling max and min stroke thickness?**
Yes.

C130. **Ref F60, selection of faces or edges will either be via the existing selection tools available in the Blender wireframe view, or by "painting" areas using the Blender Grease Pencil. If one particular method is more desirable, or if another method is required, please state it here.**
Would expect either Grease Pencil or the "Mark Seam" functionality in the mesh editor (used for indicating UV unwrapping). As a general point, [*N280: might want to consider the UV maps as a useful source of information – this is a way the user can specify the structure of the shape in more detail.*]

4.2.2 Additional Requirements

F170. **Should there be natural variation in generated stroke waviness/curvature?**
Natural variation as an optional requirement, configurable if present.

F180. **If yes to the above, should the User have control over the global variation of waviness/curvature?**
Natural variation as an optional requirement, configurable if present.

F190. **Should there be natural variation in generated stroke thickness along the length of a stroke (independent of distance from the camera)?** Yes, but optional requirement, and configurable.

F200. **If yes to the above, should the User have control over the global variation of thickness?**
Yes, but optional requirement, and configurable.

F210. **Should the User have control over the preferred global density of generated strokes?**
Yes, configuration of density is important. Perhaps only globally; ideally also locally somehow (but may not be feasible).

F220. **Should the User have control over the directionality of generated strokes? If so, how should this be controlled?**
Not clear how this could be usefully controlled? Doesn't seem a key requirement.

F230. **Should the User have control over the global stroke colour?**
Yes, though this isn't critical (since could just open in an SVG editor and recolour).

F240. **Should the User have control over the canvas (image background) colour?**
Yes, though this isn't critical (since could just open in an SVG editor and recolour).

F250. **If there are other functional requirements not captured by the sections above, please state them here.**
If supporting multi-meshes, would be good to be able to [*F250: separate strokes from objects into SVG layers so they can be edited separately.*] [*F260: Separating into layers could also be used for strokes from different sources (e.g. silhouette, contour, strokes from different light sources).*] This isn't a critical feature, but being able to easily work with groups of strokes in the SVG editing stage would be useful. Certainly, one preference would be that [*N290: editing the generated SVG files should be structured so that this is pleasant and efficient to do.*]

[*N300: Consideration of how shadows will be handled is important.*] This might be straightforward but should be considered.

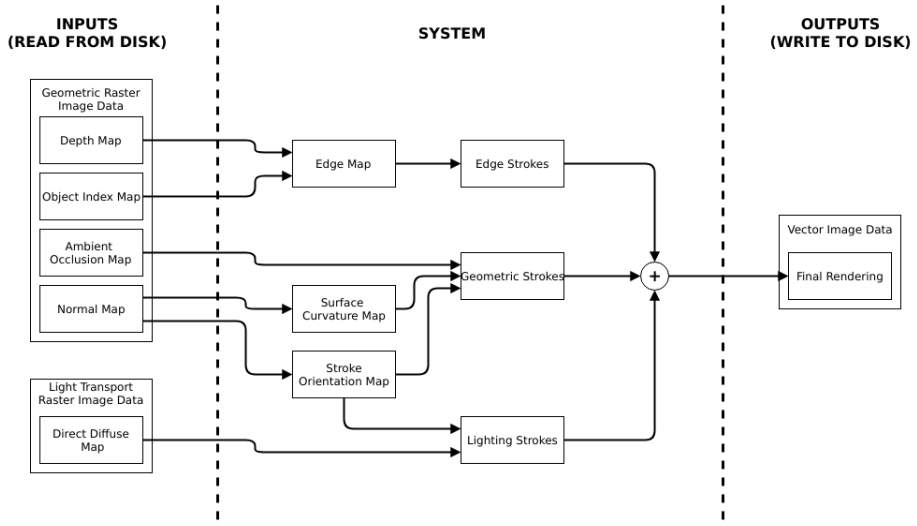
5 Assumptions

- A purely [*A10: image-space*] based approach will be taken, i.e. no inputs are taken from Blender's object-space.
- [*A20: It is not possible to obtain image-space data directly from Blender's render passes via the Python API*].
- As a consequence of A20, [*A30: Blender must save the required input images to disk before they can be processed by the System.*]
- As a consequence of A30, we define a new requirement: [*F280: The System shall automatically activate required render passes and produce the required compositor node setup for saving these images to disk.*]
- [*A40: The goal of the System is to obtain the Final Rendering for use in other software packages.*]
- As a consequence of A50, we define a new requirement: [*F290: The Final Rendering shall be saved to disk, and will not be presented on-screen within Blender.*]

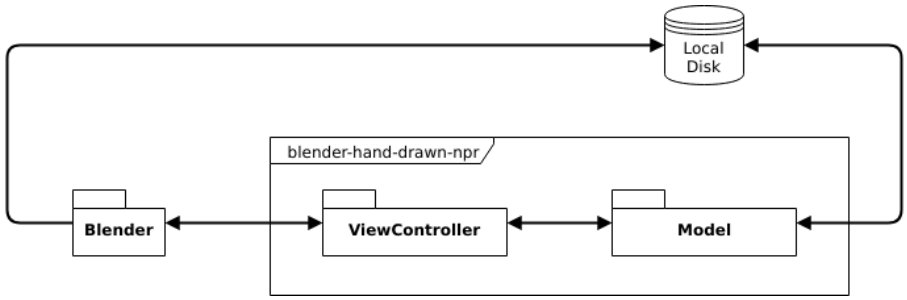
6 Proposed Design

6.1 System Overview

Adapted from Kang et al. (2006).

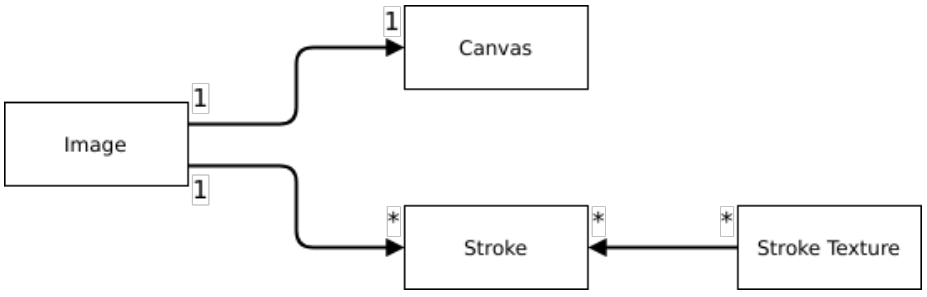


6.2 Architecture



6.3 Domain Model

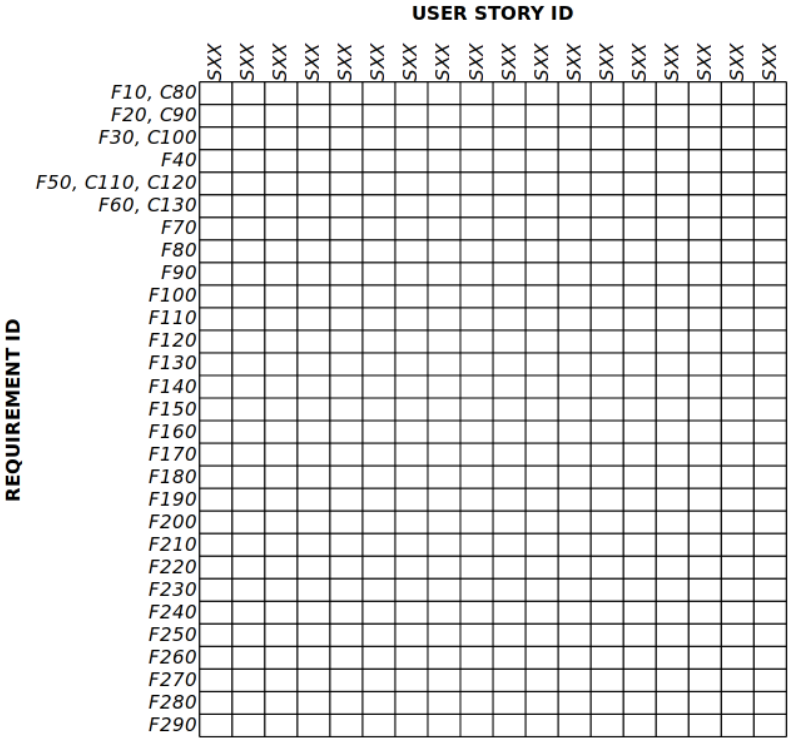
Based upon (Hertzmann, 2002), (Salisbury et al., 1994), (Salisbury et al., 1996) and (Winkenbach and Salesin, 1994)



7 Risks

8 User Stories

A Coverage Matrix



References

Hertzmann, A. (2002). Stroke-based rendering. In *Advances in NPR for Art and Visualization*.

Kang, H. W., Chui, C. K., and Chakraborty, U. K. (2006). A unified scheme for adaptive stroke-based rendering. *The Visual Computer*, 22(9):814–824.

Salisbury, M., Anderson, C., Lischinski, D., and Salesin, D. (1996). Scale-dependent reproduction of pen-and-ink illustrations. pages 461–468. ACM.

Salisbury, M., Anderson, S., Barzel, R., and Salesin, D. (1994). Interactive pen-and-ink illustration. pages 101–108. ACM.

Winkenbach, G. and Salesin, D. (1994). Computer-generated pen-and-ink illustration. pages 91–100. ACM.