

crochet craft

CrochetCraft

3D modelling for crochet patterns

Team 7 - Allen Liu, Dhananjay Patki, Jasmine Tai, Simran Thind, Osman Wong, Carol Xu

This final report is broken into the following sections:

*Introduction, Deployment Information, Final Video
Requirements Specification
Design Documentation
Verification and Validation
User Manual*

Introduction, Deployment Information, Final Video

Introduction

The following preface is a modification of the blurb written for the [capstone website](#): Crochet is the art of creating textiles using a crochet hook and yarn. To crochet an object, one needs to follow a “crochet pattern”, akin to a cookbook recipe, to create whatever is desired. However, crochet patterns do not always indicate exactly what the final work looks like. [CrochetCraft](#) is a computer-aided design (CAD) web application that aims to address this problem by allowing users to make crochet stitches digitally and see the result in three dimensions (3D). This is useful for pattern designers to iterate faster, and helpful for novice crocheters to modify patterns to suit them.

This Final Year Design Project (FYDP SE25) has been cooking since the 3B term's SE390 Design Project Planning class, where it originated from Mini Project 3 (MP3). The goal for that iteration was to produce a literature review about current research relating CAD and crochet. At this point, we found there were some active areas of crochet research that may be useful, such as: using stitch meshes for crochet, using specific graph-based representations, and applying techniques from papers that detailed how to convert some 3D shapes to crochet (opposite of our goal).

A bulk of the work happened in the co-op terms after 3B and in 4A/4B. This report will go into detail about the technical aspects and decisions for CrochetCraft, as well as present artifacts from 4A's Software Requirements: Specification & Analysis class (SE 463 - Fall 2024).

Deployment Information

CrochetCraft makes use of the following technologies and libraries: [Svelte](#) (a component-based front-end software framework, as CrochetCraft runs as a web application), [TypeScript](#) (main programming language of the project), [Skeleton UI](#) (design system), [Tailwind CSS](#) (CSS framework), [GitHub](#) (repository hosting), [Cloudflare Pages](#) (site publishing), [Three.js](#) (3D rendering via WebGL), [Jest](#) (testing framework), [yarn](#) (package management), [neodrag](#) (for easy draggable elements), [Vite](#) (build tooling), and [Flaticon](#) (user interface icons).

For code review and deployment, some continuous integration principles were implemented. For example, with [Github Actions](#) (a method to automate software development workflows), when a pull request (PR) was opened, assignees, reviewers, and appropriate tags would get added automatically to the pull request metadata. Cloudflare Pages would also generate a preview link. [CodeQL](#), GitHub's “semantic code analysis engine”, runs on every push. Lastly, project dependencies are kept up-to-date via [Dependabot](#), which opens up PRs based on security updates pulled from vulnerability databases.

As aforementioned, actual live deployment is done through Cloudflare Pages. Svelte's [adapter-cloudflare](#) is used to build the app for Cloudflare Pages (the adapter is not explicitly specified, but [adapter-auto](#) detects its environment and uses the correct adapter automatically).

Final Video

A video demonstrating CrochetCraft, as well as walking through much of the same content as this report, can be found at the root of the [CrochetCraft repository](#).

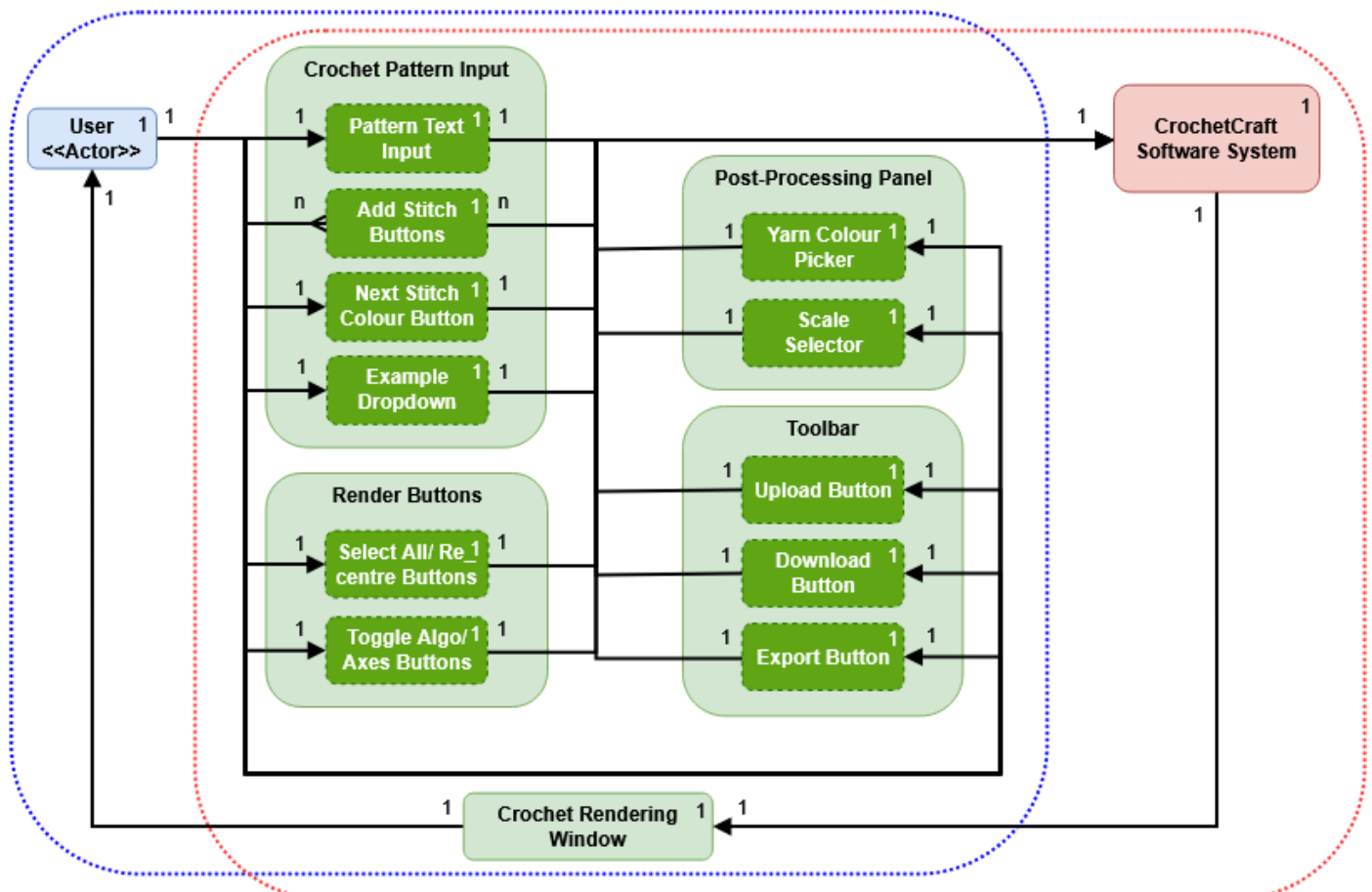
Requirements Specification

The requirements specification below is an adaptation of the specification some members of the team created when taking SE 463 - Software Requirements: Specification & Analysis back in Fall 2024. In this course, the overall specification was broken up into individual assignments. These have since been updated to reflect the final version of the project.

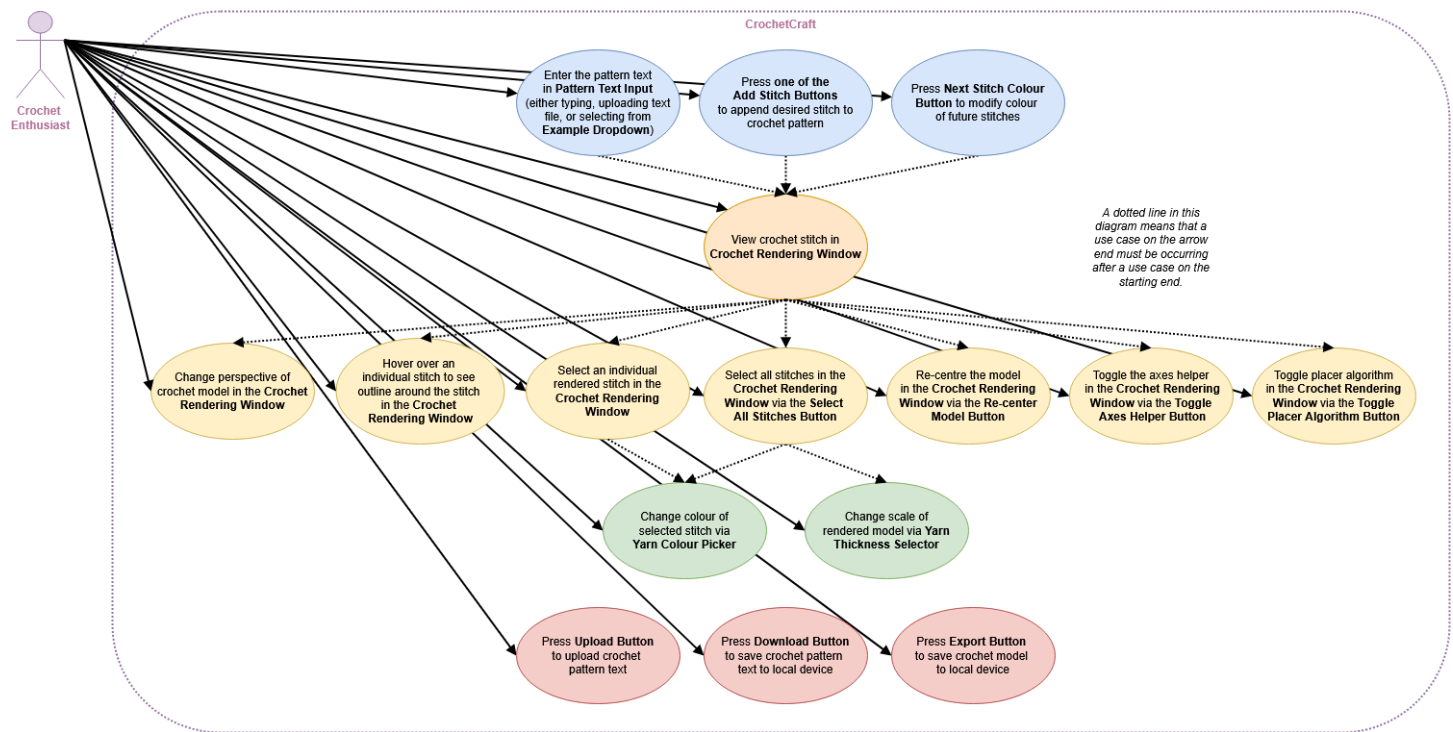
Domain, Class, and World Model (Blue is ENV, Red is SYS)

There are four major areas of input to CrochetCraft (using terminology from the model diagram - these terms may be referred to in slightly different ways in this report beyond this section):

1. the “Pattern sidebar” on the left of the UI: a text box for entering a crochet pattern; a set of **Add Stitch Buttons** where the user can enter individual stitches into the crochet pattern (e.g. **ch**); an **Example Dropdown**, where the user can select pre-formed patterns, and they populate the text box; and a colour picker (via the **Next Stitch Colour Button**) for setting the colour of the next input stitches.
2. the toolbar at the top of the UI: contains an **Upload Button** (for uploading a crochet pattern text file to the program), a **Download Button** (for downloading the pattern text that lives in the text box), and an **Export Button** (for downloading the rendered model itself in one of two formats: OBJ or glTF).
3. the “Post-Processing panel” which appears to the user when there is at least one stitch selected in the **Crochet Rendering Window**: contains the **Yarn Colour Picker** (which enables the user to change the colour of the selected stitch(es)), and if all stitches in a model are selected, the user can scale the model via the **Scale Selector**.
4. the group to the bottom-right of the **Crochet Rendering Window**: the **Select All Stitches Button**, the **Re-centre Model Button**, the **Toggle Axes Helper Button**, and the **Toggle Placer Algorithm Button**.



Use Case Model



Domain Assumptions, Exceptions, Variations

Assumptions

1. Users will not try to submit invisible characters, combining characters, or other Unicode characters that interfere with normal text display.
2. The computer running the system is powerful enough to handle the crochet pattern input, and rendered crochet model.
3. The user's computer has enough space for crochet patterns to be downloaded.
4. The user's computer has enough space for crochet models to be downloaded.
5. The user will only use a browser that supports the prerequisite technologies for the system.
6. The user will only upload valid pattern texts.
7. Crochet patterns in the real world do not contain yarn threads intersecting through each other.
8. Crochet patterns in the real world are mostly composed of a common set of well-defined stitches.
9. The stitches in crochet patterns in the real world connect to each other.
10. Crochet patterns in the real world are constructed one stitch at a time.
11. Crochet patterns in the real world are constructed, so each stitch connects to the previous one in the pattern and possibly to other even early stitches in the pattern.
12. Crochet patterns in the real world only contain one foundation stitch.
13. Yarn colours in the real world fall into a domain that can be adequately described by 24-bit RGB.
14. Users will not attempt to tamper with the system on their local device by modifying their browser, or the contents the browser is displaying.

Exceptions

1. The user inputs a pattern with an unrecognized stitch name.
2. The user inputs a pattern with invalid or unrecognized syntax.
3. The user tries to load a file that is not a crochet pattern (.txt file).
4. The user inputs a pattern with an excessively large amount of stitches.
5. The user enters an invalid value in the settings panel (e.g., a non-number when a number is required).

6. The user opens the app in an outdated/incompatible browser.
7. The user closes the window without saving the pattern being typed.
8. The user adds more than one foundation stitch.

Variations

*Change perspective of crochet model in the **Crochet Rendering Window**:*

1. The user pans the **Crochet Rendering Window** such that the stitch is no longer visible on the user's screen.
2. The user zooms into/out of the crochet model in the **Crochet Rendering Window**.
3. The user rotates out of the crochet model in the **Crochet Rendering Window**.

*Hover over an individual stitch to see the outline around the stitch in the **Crochet Rendering Window**:*

4. The user hovers over empty space instead of an individual stitch in the **Crochet Rendering Window**.
5. The user hovers over a stitch with the same colour as the background of the **Crochet Rendering Window**.

*Select an individual rendered stitch in the **Crochet Rendering Window**:*

6. The user long-presses an individually rendered stitch in the **Crochet Rendering Window**.
7. The user selects empty space instead of selecting a crochet stitch in the **Crochet Rendering Window**.

*Change colour of selected stitch via **Yarn Colour Picker**:*

- The **Yarn Colour Picker** will be represented by the default visual colour picker interface from the user's browser. This means that the presentation of the colour picker interface will vary between browsers and platforms.
- 8. The user enters in hexadecimal code to change the colour of the selected stitch via the **Yarn Colour Picker**.
- 9. The user enters in RGB to change the colour of the selected stitch via the **Yarn Colour Picker**.
- 10. The user selects their desired colour from a gradient palette to change the colour of the selected stitch via the **Yarn Colour Picker**.

*Enter the pattern text in **Pattern Text Input**:*

11. The user types in their pattern text into the **Pattern Text Input** box.
12. The user copy-and-pastes in their pattern text into the **Pattern Text Input** box.
13. The user uses speech-to-text to input their pattern text into the **Pattern Text Input** box.
14. The user uses the **Example Dropdown** to populate the **Pattern Text Input** box.

*Press **Add Stitch Button** to append desired stitch to crochet pattern:*

15. The user clicks the "chain" button to append the desired stitch to the crochet pattern.
16. The user clicks the "single crochet" button to append the desired stitch to the crochet pattern.
17. The user clicks the "increase" button to append the desired stitch to the crochet pattern.
18. The user clicks the "decrease" button to append the desired stitch to the crochet pattern.

*Toggle the axes helper in the **Crochet Rendering Window** via the **Toggle Axes Helper Button**:*

19. The user clicks this button to turn on the axes helper.
20. The user clicks this button to turn off the axes helper.

*Toggle the placer algorithm in the **Crochet Rendering Window** via the **Toggle Placer Algorithm Button**:*

21. The user clicks this button to use the iterative forces (IF) placing algorithm.
22. The user clicks this button to use the gradient descent (GD) placing algorithm.

The User Manual is located at the end of this report.

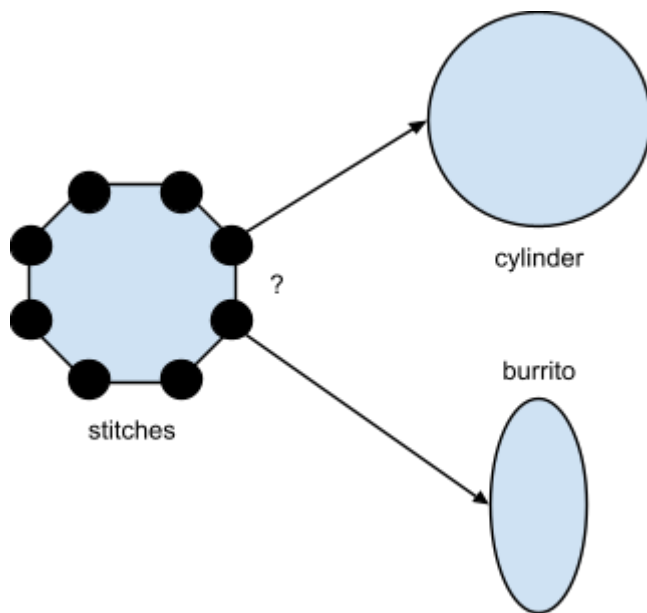
Design Documentation

This section is split into two parts: “Technical Problems”, which discusses a few of the challenges encountered when building CrochetCraft, and “Architecture”, which dives into the pipeline from user input to final 3D model.

Technical Problems

1. The “Bean-Burrito Problem”

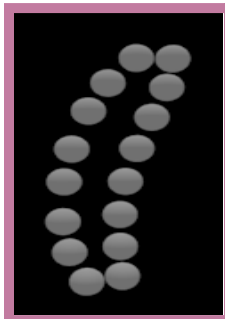
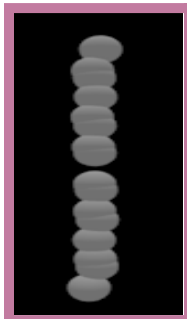
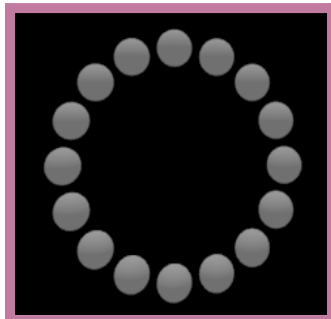
When determining stitch-level placement in 3D space, we simulate individual stitches as spheres and repeatedly apply forces between connected stitches. However, there is no single “correct” geometry for any given pattern of stitches, as the “yarn” can be bent in various ways in both the placement-stage simulation and in reality while still being the same sequence of stitches at its core. For example, consider a crochet pattern made of a stack of rings (an open cylinder). The user-expected geometry may be a cylindrical shape with a near-circular cross-section, but a “squashed”, ellipsoid resembling a burrito is also topologically correct.



In general terms, it means that there are different geometries for stuffed and unstuffed crochet models, and there is no definitive way to tell the difference based on the pattern alone.

The team first encountered this issue in 4A (specifically on 2024-11-06) during initial experiments with iterative forcing (a paradigm for placing stitches, discussed more in depth in the “Architecture” part of this section). While the simplest iterative forcing method creates reasonable results and converges relatively quickly for stitches placed “close” to their intended final pattern, a pessimistic case where stitches are poorly placed can produce unexpected results (see the images at the bottom of the page).

Our approach to solving this problem was to assume the most convex shape is the desired arrangement. Put shortly, given a “ring” of stitches, we initially place them in a circle. While there is no enforcement of convexity in the current product, a method to avoid sharp edges was proposed: For each three points (x, y, z) where x is connected to y and y to z, exert a force pushing x and z away from each other, perhaps proportional to:



$$\exp((\vec{y} - \vec{x}) \cdot (\vec{y} - \vec{z}))$$

Additional considerations for this formula include normalization of vector differences and a maximum limit on the force to prevent unbounded increases in stitch distance.

2. Mesh Performance

Three.js has a *Mesh* class, which represents “triangular polygon mesh based objects”. For posterity (per Wikipedia), a polygon mesh “is a collection of vertices, edges, and faces that defines the shape of a polyhedral object’s surface”. *Mesh* is also a base class for other classes like *BatchedMesh*, *InstancedMesh*, and *SkinnedMesh*. Currently, every stitch in a 3D model is represented by an individual *Mesh*. However, the more stitches there are in a crochet pattern to render, the more computationally expensive using *Mesh* gets, which slows down the program.

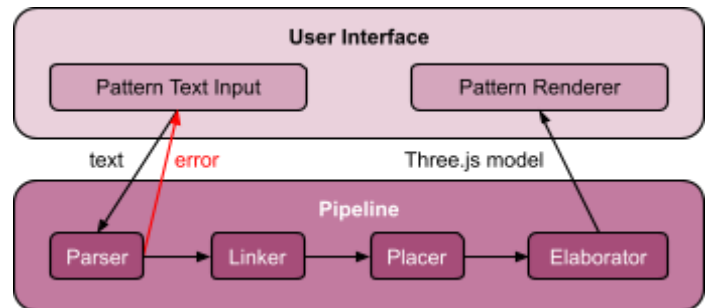
Using either *BatchedMesh* or *InstancedMesh* would aid rendering speed, as both reduce the number of draw calls necessary. *BatchedMesh* is applicable to a large amount of objects “with the same material but with different geometries or world transformations” whereas *InstancedMesh* is also useful for lots of objects, but they should have the same geometry and materials (but different world transformations). In either case, while program speed would be improved, the capability for the user to select and interact with individual stitches would no longer be a trivial task (as is the code currently).

Architecture

The process of creating a 3D model from user crochet pattern input follows this pipeline:

Parser

This component checks the validity of the user-entered crochet pattern and converts the pattern text into internal programmatic representations of stitches. You can find an example of the parsing grammar in the “User Manual” section of this report.

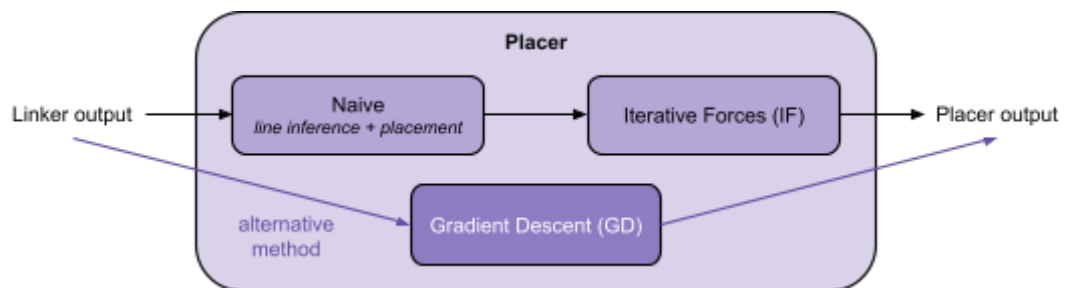


Linker

Determines connections between stitches and groups stitches into rows based on the presence of turns and the availability of previous “parent” stitches.

Placer

TL;DR: Determines the position and orientation of stitches. This starts with an initial placement step, followed by repeated application of forces between connected stitches with the goal of reaching an optimally stable configuration.



The placer takes information from the linker and places stitches approximately where they should be in the final model. The placer does not exactly care about the geometries of each stitch, only their position and orientation. There are two independent implementations of the placer, each with their strengths and weaknesses. Both work fundamentally similarly, starting by making some initial placement based on row ending type (rows ending in turns, like in a rectangle, are placed linearly, while rows that don’t end in turns, like in cylinders, are placed in perfect circles). They both then repeatedly apply forces with the goal of reaching an optimally stable configuration, stopping after a maximum number of iterations or once the change per iteration

drops below a certain threshold. The primary difference between the two implementations lies in how forces are calculated.

Iterative Forcing (IF)

The IF placer models connections between stitches as springs with a neutral position such that the “spheres” of two connected stitches will be just barely touching at a single point. Force is calculated by the deviation from that “neutral” distance for each pair of stitches using Hooke’s Law ($F = -kx$). However, “force” is something of a misnomer: the “forces” aren’t true forces in the physics sense, but rather an indication of the displacement during the next time step. (It is a term that is mainly used as the spring metaphor is an intuitive way to think about it.) As such, the “spring constant” is chosen as 0.5, such that two connected stitches with no other connections will return to a neutral position in just one iteration.

The IF placer’s handling of repeated rings warrants separate discussion, as repeated rings of different sizes pose a question of placement. A cone shape, a circle, and a cylinder all consist of many circular rows, but the vertical arrangement between each of them differs. To handle this, if the difference in radius is less than a full stitch width, the “outer/upper” row is placed lower than it otherwise would be, such that the two rings would just barely be touching. Essentially, it tries to fit circles within each other as much as their radii will allow. This is why the bowl example pattern (see UI) is flat-bottomed when using IF placer, as many of the rings can nearly fit inside each other.

Gradient Descent (GD)

The GD placer is quite similar to the iterative forcing placer, but can be thought of differently. Instead of using forces to attract and repel, we use an “error” measurement. The placer has one goal: to minimize the error of the placed stitches.

Consider a particular **placement** (output of the placer) represented as a vector \vec{P} (conceptually, it might contain the coordinates of each stitch, for example). Then, the **error** of the placement is a function $E(P)$ which produces a nonnegative scalar. Our goal is to minimize $E(\vec{P})$.

The heart of the algorithm is the E function: how the error is calculated. The error is the sum of each **constraint error**. For each pair of stitches, there can be zero or more **constraints** between them. Each constraint contributes some amount to the overall error.

For example, a stitch may have a constraint with its previous stitch, which states that the stitches should be 1 unit away from each other. Then, if the two stitches are at positions \vec{s}_1 and \vec{s}_2 , then the constraint error might be calculated as $(\|\vec{s}_1 - \vec{s}_2\| - 1)^2$.

Notice how, when the stitches are exactly 1 unit apart, the constraint error is zero. Otherwise, the error becomes larger as the distance between the stitches moves away from the “ideal” of 1 unit.

The GD placer starts by making an estimate of the placement, like the IF placer, and this becomes the initial guess \vec{P}_0 . Then, it iteratively improves the guess via gradient descent: $\vec{P}_i = \vec{P}_{i-1} - \alpha \frac{dE(\vec{P}_{i-1})}{d\vec{P}_{i-1}}$.

Here, α is the speed of the gradient descent; with a lower α , the result is more accurate, and with a higher α , it converges after fewer iterations. Currently, the way the code is actually implemented doesn’t compute the error; rather, it tries to (crudely) compute the derivative directly.

Furthermore, as an optimization (which may compromise on accuracy), each component of \vec{P} is updated one after the other, rather than updating the entire vector in one step. These implementation details will likely be changed in the future.

IF vs GD Comparison

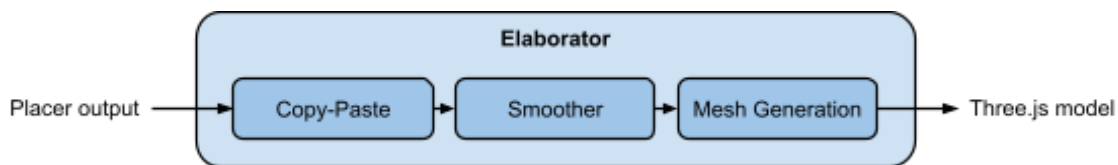
IF placer produces a more intuitive result for linear patterns such as rectangles, as the straight linear placement generally persists through the force iteration. GD placer's output for linear patterns, while not technically incorrect, tends to introduce curves where the user might not expect. For example, a long chain (with a pattern like 'ch 50') will be perfectly straight with IF's placement, but will curve with GD placer.

On the other hand, circular patterns are handled more gracefully by GD placer, particularly when it comes to concentric circles (i.e. not including the cylinder, which is handled well by both). IF placement has a “bumpy” result, while GD's placement, while curved on a large scale, is smoother at a local level.

Orientation Setting

After the placement concludes for either placer implementation, orientation information is calculated for each stitch based on the locations of its neighbours and its parents. These are stored as rotation quaternions representing the rotation needed, assuming initial orientation is along the positive x-axis. This orientation information is used to rotate the base models in the elaboration stage.

Elaborator



TL;DR: The elaborator constructs yarn-level geometries for each stitch based on preset base models of stitch types, producing a final model to be rendered using Three.js in the UI. A smoothing process is performed at the border between stitches to ensure the resulting “yarn” has a continuous, nice appearance.

The elaborator takes the position and orientation information of stitches and generates the actual curve geometry corresponding to the real stitch. Each stitch type also has an associated “base model”, which is a preset 3D model of that stitch type to be used as a building block for the final crochet pattern.

Copy-Paste

For each stitch, the base model is copy-pasted into the stitch's placed position and then rotated based on the orientation calculated in the placer. The base models are currently represented as a sequence of cubic Bézier curves, which are easy to model in external programs (i.e. Blender), and to use within THREE.js.

Smoothing

After stitches are copy-pasted in, a smoothing process is done to ensure that continuity and a smooth yarn appearance is maintained. As the base models are made to be smooth curves to begin with, this mainly concerns the start and ending points (i.e. where the yarn “transitions” between stitches).

The smoother currently maintains up to first derivative smoothness by moving the starting point and first anchor of each stitch. Continuity is achieved by simply moving the starting point onto the end of the previous stitch. Derivative continuity is done by projecting the first anchor point onto the line that the previous stitch's last point and anchor lie on. In the case that the projection would invert the stitch, the projection is inverted to

maintain smoothness, and avoid creating a cusp. Since the slope of a Bézier curve approaches this line at the very end, this results in a smooth connection between stitches.

Interpolated Shearing

A third step that was not integrated in the current product is the application of shearing to each stitch based on the relative position of connected stitches. The goal of this interpolated shear is to produce a more realistic look in the final stitch geometry. This would require information about linked stitches for each stitch and the relative positions of those linked stitches.

Tip-Loop Substitution

A proposed method to ensure that the various loops in each stitch goes through the correct spot in its parent and/or neighbour stitches is: to store additional information about the position of the tip of each stitch type's loop and the position it expects the neighbour or child stitch to loop through. A pass would be done to replace the point in each stitch's Bézier curve representing the end ("tip") of the loop with the relative position where the next stitch expects the loop to pass through. A similar process could be done with stitches looping through/around parent stitches. This has thus far remained a theoretical approach, as well-made base models have avoided significant yarn-level collisions for most patterns within the current scope of the project.

Verification and Validation

As aforementioned in the previous “Deployment Information” section, as far as checking for user interface issues goes, PRs having deploy previews automatically generated helped with functionality and some system/acceptance testing. However, we did have our own other methods for conducting unit testing (testing individual components and functions) and end-to-end testing (E2E - system testing in its entirety).

Unit Testing

In order to validate that the parser and linker (phase 1 and 2 of the end-to-end pipeline, respectively) worked as intended for all cases, [Jest](#) tests were created. Jest is a JavaScript/TypeScript testing framework with a robust API (application programming interface) that works across a variety of frameworks, ensuring that testing is both fast and efficient.

Two main types of tests were created for both the parser and linker:

1. The first type was simple base test cases - these were heavily used in the development process to ensure that work done on these parts of the pipeline actually produced correct functionality.
 - An example of a base test case for the parser would be being able to parse just a single stitch, such as a single crochet:

```
describe('simple tests', () => {
  test('single stitch', () => {
    expect(parse('sc')).toEqual(slkt(st(StitchType.Single, 1)));
  });
});
```

...
2. The second were edge case tests - the primary purpose of these test cases was to try scenarios that exercised a specific capability of the pipeline component.
 - For example, the parser needed to handle parts of a crochet pattern that indicated a loop, and therefore several test cases were created which contained contrived loop scenarios that would test the limits of the code. While many of these scenarios would be very unlikely to occur in actual user-provided input, these tests ensured correctness across all valid user inputs.

End-to-End (E2E) Testing


For E2E testing, a special page under the path `/experiments/e2e-testing` was made. On this page, one of several preset patterns could be selected (although raw text entry was omitted on this page), and the pattern text would be run through the entire pipeline and rendered as it would be on the main page. Switching between the iterative forcing (IF) and gradient descent (GD) placers was also supported on this E2E testing page, even before the decision was made to allow both modes on the main UI. Finally, the last step of the pipeline, the elaborator, could be toggled. When turned off, the stitches were rendered as spheres with lines indicating connections, allowing for both easier debugging of placer-related changes and manual testing of the placer before the completion of the elaborator.

Additionally, other subpages were made for special experiments to test out specific algorithms or features. These tests included a testing ground for the force iteration of the IF placer, experiments related to 2D and 3D shearing (per Wikipedia - “a shear mapping is an affine transformation that displaces each point in a fixed direction by an amount proportional to its signed distance from a given line parallel to that direction”), and initial demos of rendering simple objects using THREE.js. These experimental pages allowed us to test out specific parts of the pipeline without requiring the completion of other items - essentially, manual unit testing before deciding on use and integration.

User Manual

The user manual presented on the next page is an adaptation of the user manual some members of the team created when taking SE 463 - Software Requirements: Specification & Analysis back in Fall 2024.


Addendum: FYDP poster



CrochetCraft

3D modelling for crochet patterns

Team 7 - Allen Liu, Dhananjay Patki, Jasmine Tai, Simran Thind, Osman Wong, Carol Xu (SE25)



UNIVERSITY OF
WATERLOO

FACULTY OF
ENGINEERING

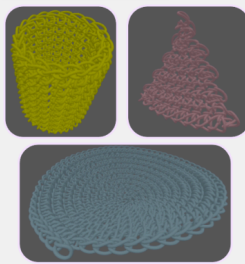
MOTIVATION

- Popular craft:**
 - Per the Association for Creative Industries, 28.8 million Americans participated in knitting/crochet in 2016.¹
- Lack of tooling:**
 - There is a glut of research and products related to computational knitting, while there is little for crochet.²
- Crochet is fun!**

OBJECTIVE

- Provide a CAD app:**
 - Build a "computer-aided design" (CAD) web application that allows users to input a text crochet pattern, and receive a generated 3D object based on the pattern.
- Cater to different users:**
 - Facilitate faster iteration for pattern designers and experienced crocheters.
 - Allow novice crocheters to modify patterns to suit them.

RESULTS



Many cool shapes!

USER INTERFACE

(Menu) Toolbar:

Users are able to upload and download patterns as .txt files, plus export their 3D object in OBJ and glTF.

Editor Sidebar:

There are multiple input methods: typing directly into the pattern textbox, clicking the stitch buttons, or loading an example.

There is also indication for errors, along with their location.

Post-Processing Panel:

Post-render, users can select and change the colour of individual stitches. Selecting all unlocks a resize slider.

Render Canvas:

3D model is displayed here. Users can zoom and pan around the model.

Render Toolbar:

- Select all stitches button
- Re-centre model button
- Toggle visibility of axes helper
- Toggle placer algorithm

3D PIPELINE

User input from UI

ch 20

Parser

Linker

Placer

Elaborator

Render to the UI

Checks validity of user-entered stitch pattern and converts pattern text into internal programmatic representations of stitches.

Determines connections between stitches and groups stitches into rows based on the presence of turns and the availability of previous "parent" stitches.













Determines position and orientation of stitches. This starts with an initial placement step, followed by repeated application of forces between connected stitches with the goal of reaching an optimally stable configuration. There are two independent but similar implementations which the user can choose between: IF (Iterative Forcing) and GD (Gradient Descent).

Constructs yarn-level geometries for each stitch based on preset base models of stitch types, producing a final model to be rendered using Three.js in the UI. A smoothing process is performed at the border between stitches to ensure the resulting "yarn" has a continuous, nice appearance.

ALTERNATIVES

- Rendering alternatives:**
 - Rendering alt. included *Unity* or *WebGL* with shaders.
 - Decided on *Three.js* for its maturity and ease of use for both geometry construction and rendering.
- Rendering speed:**
 - Use of *InstancedMesh* could reduce stitch geometry construction overhead, but would complicate stitch selection and interaction.

SOFTWARE INFO

- site: crochetcraft.jtai.ca
- code size: 213 KiB
- technologies used:
 -    
 -    
 -    

REFERENCES

- Decker, I., *Knitting and Crochet Today: Statistics, Trends, and More*. Anthony Thomas Advertising Agency, May 9, 2018. <https://blog.anthonymthomas.com/knitting-and-crochet-today-statistics-trends-and-more>
- Chen, M. B., and Edelstein, M. *Amigurumi Crochet Patterns from Geodesic Distances*. Bridges 2024 Conference Proceedings (Computer Science Department, Technion - Israel Institute of Technology). <https://archive.bridgesmathart.org/2024/bridges2024-369.pdf>

Many thanks to Professor Christopher Batty of the University of Waterloo Computer Graphics Lab for his insight, as well as Director Victoria Sakhnini, Associate Director Paul Ward, Program Manager Angie Hildebrand, and the rest of the Software Engineering Office for their support. March 2025.

CrochetCraft User Manual

Simran Thind

Jasmine Tai

Carol Xu

Dhananjay Patki

Allen Liu

Osman Wong

April 16, 2025

Contents

1	Introduction	2
1.1	Product Overview	2
1.2	First Sample Run	2
2	Conventions	4
2.1	User Assumptions	4
2.2	Computer System Assumptions	4
2.3	Notational Conventions	4
2.4	Terms	5
2.5	Other Abbreviations	7
2.6	Basic User Interface Goals	7
2.7	Organization of this Manual	7
3	Use Cases	8
3.1	Crochet Pattern Entry	8
3.2	Viewing Crochet Pattern	9
3.3	Interacting with the Rendered Model	10
3.4	Post-Processing the Rendered Model	10
3.5	Loading and Saving Crochet Patterns	10
4	Troubleshooting & Tips	12
4.1	Troubleshooting	12
4.2	Tips	13
5	Limitations	14

Chapter 1

Introduction

1.1 Product Overview

Crochet is a method to turn yarn into textiles. It is a form of art that uses a crochet hook to knot the yarn in such a way to produce a three-dimensional product, which can range from clothing and toys to bags and public art displays. A crochet pattern is a list of steps to produce a specific product, analogous to a cooking recipe. The design of crochet patterns typically involve many iterations of producing, refining, and experimenting, and the CrochetCraft app attempts to simplify this process.

CrochetCraft is a web application that produces real-time 3D visualizations of crochet patterns. It allows designers to input their textual pattern and view the simulated product resulting from the pattern. Designers can refine and iterate on their pattern without the time-consuming process of constructing the design in the real world each time.

1.2 First Sample Run

Start the CrochetCraft application by opening `crochetcraft.jtai.dev` in your web browser. Once loaded, the application's screen layout will be displayed, as shown in figure 1.1. Left-click anywhere in the pattern text box, and type or copy-and-paste the following pattern:

```
0. ch 11  
1. sc 10  
2-10. ch 1, turn, sc 10  
ch 1
```

After all of the above text has been inserted, a 3D visualization of the pattern will be displayed in the rendering area. Next, use the mouse to highlight the text 2-10 in the Pattern text box, and overwrite it by typing 2-20. The modified pattern will be visualized in the rendering area.

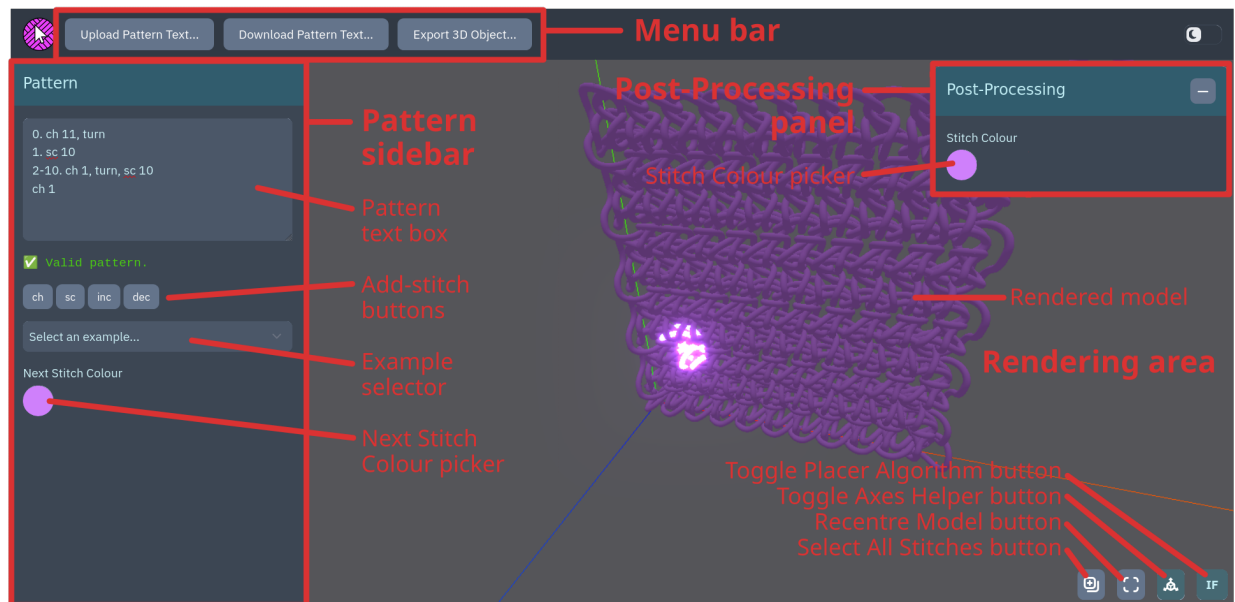


Figure 1.1: The screen layout of CrochetCraft

Chapter 2

Conventions

2.1 User Assumptions

You, the user of CrochetCraft, are assumed to understand American crochet terminology, and be competent in the basic usage of the device that is running CrochetCraft.

2.2 Computer System Assumptions

CrochetCraft runs in a web browser, so it is assumed that you are using a browser that is up-to-date, and has not been tampered with. It is assumed that you are using a device with a keyboard and mouse. If you wish to use any features of the computer filesystem, it is assumed that your device has sufficient free space for these features.

2.3 Notational Conventions

The following text conventions are used in this manual:

- Libertine is used for normal text.
- *Libertine italicized* is used for new terms in normal text.
- **Libertine bolded** is used for emphasis.
- Noto Sans is used for program names and user interface input/output.
- Noto Sans italicized is used for widget names.
- Bitstream Vera Sans Mono is used for internal representation contents.
- *Bitstream Vera Sans Mono italicized* is used for variables of internal representation syntax.

2.4 Terms

Listing 1 details the basic grammar used when parsing the pattern text. Furthermore, table 2.1 contains the list of keywords. In each row, the keywords listed next to each other can be used interchangeably. For example, `sc2tog` has the same meaning as `dec`, so whenever the grammar includes “`dec`”, `sc2tog` can also be used.

```

pattern := (line? NEWLINE)* line?
line := row_number? foundation? instructions

row_number := INTEGER (“-” INTEGER)? “.”
foundation := “mr” “,”

instructions := (instruction “,” | COLOR “:”)* instruction
instruction := “turn” | stitches | repeat

stitches := stitch | stitch count | count stitch
stitch := “dec” | stitch_type “st”? “inc”?
stitch_type := “ch” | “sc”

repeat := (“ instructions “) count
          | count (“ instructions “)
          | “*” instructions “,” “rep” count “from”? “*”
          | “*” instructions “,” “rep” “from”? “*” count

count := “x”? (INTEGER | “twice” | “thrice”) “x”? “more”? “times”?

```

Listing 1: The pattern grammar understood by CrochetCraft

The following terms are used throughout the manual:

CrochetCraft the name of the web application

user the person who uses CrochetCraft, addressed by “you”

device whatever electronic device CrochetCraft is currently running on, inclusive of hardware and operating system

crochet the art of creating textiles by using a crochet hook and yarn

yarn the material used in the art of crochet

amigurumi the Japanese art of knitting or crocheting stuffed toys out of yarn

pattern a set of textual instructions that detail how to create a crochet project

stitch a simple, indivisible knot built on top of previous stitches, used to build patterns

foundation the initial loop used to start a pattern

Keywords	Meaning
stitch, st	stitch
mr, mc	magic ring
chain, ch	chain
single, sc	single crochet
increase, inc	increase
decrease, dec, sc2tog	decrease
turn	turn
repeat, rep	repeat
from	from
twice, two	two times
thrice, three	three times
times, time, x	times
more	more

Table 2.1: The keywords and their meanings, as recognized by CrochetCraft.

screen layout the buttons, text boxes, widgets, and other elements visible on the screen in the CrochetCraft app

rendering area the area of the screen used for displaying the rendered model

rendered model the 3D object created and shown based on the inputted pattern text

session an invocation of CrochetCraft

Pattern sidebar the user interface widget to the left of the rendering area

Pattern text box the text box within the Pattern sidebar which holds your input pattern text

add-stitch buttons the set of buttons below the Pattern text box that allows the user to input stitches one at a time

Next Stitch Colour picker the colour picker below the add-stitch buttons which determines the colour for the next added stitch

Post-Processing panel the user interface widget on the right side of the rendering area

Stitch Colour picker the button in the Post-Processing panel which can change the colour for the selected stitch in the rendered model

menu bar the user interface component above the rendering area, which contains the Upload Pattern Text button, Download Pattern Text button, and Export 3D Object button

Upload Pattern Text button the button in the menu bar that, when pressed, allows you to select a text file from your local device such that the contents of that text file populate the Pattern

text box

Download Pattern Text button the button in the menu bar that, when pressed, allows you to download your pattern text as a text file to your local device

Export 3D Object button the button in the menu bar that, when pressed, allows you to download your rendered model as an OBJ file to your local device

2.5 Other Abbreviations

internal representation any kind of data, not visible to the user, used internally by CrochetCraft to represent crochet patterns

GUI Graphical User Interface

OBJ a geometry definition file format, to which a rendered model can be exported

what you see is what you get (WYSIWYG) software that displays content in a realistic form while it is being edited

2.6 Basic User Interface Goals

CrochetCraft aims to be a friendly GUI for crochet amateurs and enthusiasts. Not much crochet modelling software exists, and most of this software does not provide fast and accessible rendering. The main appeal of CrochetCraft is from the benefits of its WYSIWYG approach and its capability of fast pattern iteration. Its basic goals are:

- It is simple for you to import your crochet pattern text into the Pattern text box, and it is simple for you to export a 3D object of the rendered model in OBJ format.
- It is easy for you to view your rendered model at many different angles.
- It is easy for a pattern designer to iterate and make modifications to a pattern, to improve it.

2.7 Organization of this Manual

The remainder of this manual is organized primarily on use cases. The Use Cases chapter describes the basic use cases, including the possible GUI interactions in depth. After the Use Cases chapter, the Troubleshooting & Tips chapter describes what you should do if you encounter common errors as well as how to use CrochetCraft effectively. Lastly, the Limitations chapter describes the restrictions on the current version of CrochetCraft.

Chapter 3

Use Cases

The use cases of CrochetCraft are classified by the section of the GUI your initial interaction occurs in.

3.1 Crochet Pattern Entry

Crochet patterns may be entered in two ways: using the Pattern text box or the add-stitch buttons.

You may enter the pattern text in the Pattern text box:

1. Any text may be typed into the Pattern text box, but only text that satisfies the grammar shall be considered valid.
2. If the resulting text within the Pattern text box is considered to be a valid pattern, the rendering area will update to display a new rendered model corresponding to the updated pattern text.

You may consider alternative methods of direct text entry into the Pattern text box:

- You may copy and paste text into the Pattern text box. This is done using the copy or paste shortcuts of the underlying device.
- You may use speech-to-text to input the pattern text into the Pattern text box box.

Regardless of the method of text entry, the rendering area will update to display a new rendered model corresponding to the updated pattern, if it is valid.

You may add one stitch to the end of a pattern using one of the add-stitch buttons. Each add-stitch button is labelled with the stitch that it will add to the rendered model, and each button corresponds to a different stitch.

1. You press the appropriate add-stitch button. For example, to add a chain stitch, press the button labelled ch.
2. The text corresponding to that stitch is added to the end of the Pattern text box.

3. If the resulting pattern text is valid, the rendering area will update to display a new rendered model corresponding to the updated pattern text.

As an alternative to traditional text input, you may press any of the add-stitch button multiple times to add multiple stitches. After each button press, the Pattern text box will update, and if the pattern text is still valid, the rendering area will update too.

In the case that the resultant text in the Pattern text box is invalid, the rendering area will not update, and will continue to display the same rendered model corresponding to the last valid pattern text.

3.2 Viewing Crochet Pattern

You may view the rendered model in the rendering area. Typically, no additional input is required to view the rendered model; the rendered model will be readily displayed on the screen under normal usage in the absence of your input.

You may interact with the rendering area using your mouse and keyboard to change the perspective that the rendered model is viewed from.

Panning the camera:

1. You may pan the camera by either (1) holding right-click, or (2) holding Shift and left-click, and then moving your mouse in a direction within the rendering area.
2. The rendering area's perspective will shift in the direction that the mouse moves in.

Rotating the camera:

1. You may rotate the camera by holding left-click and moving your mouse in a direction.
2. The rendering area's perspective will rotate around the centre of the rendered model along the direction in which the mouse moves.

Zooming the camera:

1. You may zoom the camera in or out by scrolling the mouse wheel up or down.
2. The rendering area's perspective will zoom in or out depending on the scroll direction. Scrolling down zooms the camera out, and scrolling up zooms the camera in.

In the case that the rendered model is no longer visible in the rendering area after the viewing perspective is changed, CrochetCraft does not correct this. However, you can reset the camera to ensure the rendered model is visible again.

Resetting the camera:

1. Click the Recentre Model button at the bottom-right corner of the rendering area.
2. The rendering area's perspective will be reset, nullifying any panning, rotating, and zooming done previously.

3.3 Interacting with the Rendered Model

You may interact with the rendered model using your mouse to either hover or select the individual stitches of the rendered model.

Hovering:

1. You may hover over an individual stitch by moving the mouse pointer so it is over one of the stitches within the rendered model.
2. The rendering area will update, so the stitch that is under the mouse pointer will glow.

If the mouse pointer is over empty space, and there is no stitch under the pointer, no action is taken, and the rendering area will not change what is displayed.

Selecting:

1. First, a stitch in the rendered model must be hovered over.
2. You may then left-click to select the hovered stitch.

In any use case involving a selected stitch — in particular, interactions with the Post-Processing panel — the stitch involved in step 2 is the one that will be affected.

If a long press is performed, the hovered stitch will only be selected if the mouse pointer still lies on top of the stitch in the rendered model when the left-click is released.

If no stitch is being hovered over, but a left-click is performed within the rendering area, CrochetCraft deselects the selected stitch, if there is one.

3.4 Post-Processing the Rendered Model

After a stitch has been selected, as described in the Interacting with the Rendered Model section, the Post-Processing panel will be displayed. This can be used to alter a stitch's colour after rendering.

It is important to note that post-processing changes only affect the rendered model and any exported 3D objects. The pattern text is not affected. This means, when the pattern text is changed, all changes made in the Post-Processing panel are lost.

Changing the colour of a stitch:

1. After selecting the desired stitch, left-click the Stitch Colour picker.
2. In the resulting dialog, select the desired colour. The dialog may vary depending on your device's operating system or web browser.

3.5 Loading and Saving Crochet Patterns

You can save your crochet pattern into a file on your device and load it later. This allows you to resume your work at a later time or share your work with others.

Saving your crochet pattern:

1. Left-click the Download Pattern Text button in the menu bar.
2. The current contents of the Pattern text box will be saved as a file onto your device. Depending on your device's operating system or web browser, the file may be saved in a standard location, or you may be prompted for a location to save the file.

Loading a saved crochet pattern:

1. Left-click the Upload Pattern Text button in the menu bar.
2. In the dialog box, select the file which was previously saved with the download function. The dialog box may vary depending on your device's operating system or web browser.

In the case that the selected file is not a valid file that was previously saved with the download function, the error File format not recognized appears, and the file is not loaded.

You can export the rendered pattern as a 3D model file, which can be opened in third-party applications.

Exporting your crochet pattern:

1. Once the rendering area displays the desired 3D model, left-click the Export 3D Object button in the menu bar.
2. The 3D model currently displayed in the rendering area will be saved as a file onto your device in OBJ format. Depending on your device's operating system or web browser, the file may be saved in a standard location, or you may be prompted for a location to save the file. The file can be opened in a variety of third-party software that supports loading files in OBJ format. Please consult the user manual for the third-party software to ensure it is capable of loading such files.

Chapter 4

Troubleshooting & Tips

4.1 Troubleshooting

You input a pattern with an unrecognized stitch name.

- Modify the pattern to change or remove the unrecognized stitch name. The recognized stitches are listed in the Terms section.

You input a pattern with invalid or unrecognized syntax.

- Modify the pattern to conform to the grammar laid out in the Terms section.

You try to load a file that is not a crochet pattern.

- Open a plain text file instead, or open the desired file with the appropriate software and copy and paste the pattern text into the CrochetCraft application.

You input a pattern with an excessively large amount of stitches.

- Reduce the number of stitches in the pattern. See the Limitations chapter for more details.

You enter an invalid value in the settings panel. For example, a non-number when a number is required.

- Enter a valid number without spaces, thousands separators, or other symbols.

You open the app in an outdated/incompatible browser.

- Use a device that satisfies the requirements listed in the Computer System Assumptions section.

You close the window without saving the pattern being typed.

- Ensure you have used the Download function to save your work before closing the web browser window.

You add more than one foundation stitch.

→ Remove the additional foundation stitches. CrochetCraft is only designed to render one crochet part at a time; different parts which are intended to be sewn together must be rendered separately.

4.2 Tips

- The example selector can be used to load a variety of sample crochet patterns, which can help you get started.
- Amigurumi patterns typically start with a magic circle, while clothing and similar textiles typically start with a slip knot and chain stitches.
- Use the Download Pattern Text button to avoid losing your work.

Chapter 5

Limitations

These are the limitations that apply to the pattern text:

- The crochet pattern text input is not inclusive of all terminology recognized by American crochet terminology, or other natural language. The crochet pattern text input is limited to what is included in the table describing the grammar used by the Pattern text box.
- Supported crochet pattern text is limited to the list of acceptable tokens only. Invisible characters, combining characters, or other Unicode characters that interfere with normal text display may cause parsing errors.

Limitations on the crochet pattern text input apply to uploaded pattern texts too.

For performance reasons, there are limitations on the number of stitches that CrochetCraft can render. CrochetCraft supports patterns with up to 20 000 stitches before user interactions have unreasonable delays of above 300 ms.

The CrochetCraft application makes a few assumptions about real-world crochet patterns, which limits what crochet patterns can be rendered satisfactorily. Here are the following assumptions:

- Crochet patterns only contain one foundation stitch.
- Crochet patterns do not contain yarn threads intersecting through each other.
- Crochet patterns are composed of a common set of well-defined stitches.
- Crochet patterns are constructed one stitch at a time.
- Each stitch connects to the previous one in the pattern and possibly to other earlier stitches in the pattern.

Index

- amigurumi, 5
- button
 - add stitch, 6, 8
 - download, 7, 11
 - export, 7, 11
 - upload, 6, 11
- camera
 - panning, 9
 - resetting, 9
 - rotating, 9
 - zooming, 9
- colour, 10
- colour picker
 - next stitch, 6
 - stitch (post-processing), 6
- crochet, 2, 5
- CrochetCraft, 2, 5
- device, 5
- foundation, 5, 12, 14
- grammar, 5
 - keywords, 5
- GUI, 7
- gui, 7
- internal representation, 7
- layout
 - screen, 6
- menu bar, 6, 11
- OBJ file, 7, 11
- panel
 - post-processing, 6, 10
- pattern, 2, 5, 14
 - entry, 8
 - exporting, 11
 - invalid, 9
 - limitations, 14
 - loading, 11
 - saving, 11
 - sidebar, 6
 - text box, 6, 8
- rendered model, 6, 9, 10
 - hovering, 10
 - interaction, 10
 - selecting, 10
- rendering area, 6
- session, 6
- sidebar
 - pattern, 6
- stitch, 5
- user, 5
- WYSIWYG, 7
- wysiwyg, 7
- yarn, 5