CIS6600: Advanced Topics in Computer Graphics and Animation

KNIT-IT:

AUTHORING TOOL DESIGN DOCUMENT

Krystal Kim & Monique Legaspi

Based on:

Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail.

Yuksel, C., Kaldor, J., James, D., Marschner, S. (SIGGRAPH 2012)

PROJECT SUMMARY

Recent yarn-based simulation techniques permit realistic and efficient dynamic simulation of knitted clothing, but producing the required yarn-level models remains a challenge. The lack of practical modeling techniques significantly limits the diversity and complexity of knitted garments that can be simulated.

The goal of this project is to design a modeling-based tool that will allow artists, animators, and game designers to quickly create yarn-level detail models of knitted garments. For example, an artist can use this tool to easily create a knitted sweater, scarf, hat, etc. for a character model. Given an input polygonal mesh, our authoring tool will first create a finer mesh representing the layout of stitches in the garment. By manipulating this mesh and assigning stitch types (knits, purls, yarn-overs), the tool can replicate a variety of complicated knitting patterns, as featured in the Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail paper. The tool will output a knitted version of the input garment mesh.

More specifically, the user will be able to change the parameters and stitch types that go into creating the polygon mesh. Given a polygonal mesh that defines the garment's surface, the user will be able to label the components of the mesh to specify the knitting direction over the surface. In real-world terms, this denotes where one would start knitting this garment and in which direction the knitter would continue knitting until the garment is completed. Using these knitting directions and the parameters specified by the user, the tool will create a high-resolution stitch mesh. The stitch mesh serves as an abstract representation of a yarn-level cloth model. Each face of the stitch mesh corresponds to a particular stitch of the yarn-level model. The stitch type of a face can be assigned by the user from a collection of predefined stitch types.

Our Alpha Version is set to be completed by Monday, March 27th, 2023, and our Beta Version is set to be completed by Wednesday, April 12th, 2023. The Final Demo Version is set to be completed by Wednesday, May 3rd, 2023.

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem or Production/Development Need

Although there have been significant advancements in the more general area of cloth rendering and simulation, there still leaves much to be desired for more complicated garments, such as those which are knitted. The inherently complex fabrication process of knitted garments results in equally complex dynamics and self-interaction, meaning yarn-level replication is necessary to achieve realism. Despite not tackling the actual dynamics of knit garments in our authoring tool, we strive to make it easier for artists and animators to create knit garments that are visually realistic and accurately constructed, so that they may later animate these garments with ease.

Before the authoring of the paper upon which our tool is based, yarn-level rendering and simulation had not yet reached a level where it could be both practical and accurate. Manual modeling, while able to ensure accuracy, was too time-consuming to be worth the effort. Generating stitch patterns over a mesh via texture-like synthesis gave the appearance of realism with relatively minimal effort, but was too general to be able to create the proper connections between threads, resulting in a garment that usually unraveled come simulation time. Simulating real-world knitting practices was also proposed, but at the time (and possibly still now), such a thing would have been far too expensive to simulate and would have been difficult for an artist to use and predict the shape of its output.

With our authoring tool, the artist can have the best of all worlds: they create the shape of the final product in the form of a mesh and using our authoring tool, they may quickly and easily transform the mesh into a visually-realistic, accurately-knitted garment that can be animated without unraveling.

1.2. Technology

Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail by Yuksel et al.

This paper proposes a new modeling technique that builds yarn-level models of complex knitted garments for virtual characters. It takes a polygonal mesh of the garment as input, and the interactive tool creates a finer mesh (a.k.a. stitch mesh) from the polygonal mesh. By manipulating the mesh and assigning different types of stitches to the faces of the mesh, the user is able to create a variety of knitting patterns. The stitch mesh is then run through a physical simulation in order to move the yarn into a realistic shape while preserving the structural integrity of the overall garment. This process creates yarn-level models of knitted clothing with intricate patterns that would otherwise be difficult to create using more traditional methods.

We chose this paper because, in the real world, knitting is a fun, creative endeavor that helps reduce stress. There are numerous things you can knit–scarves, sweaters, blankets, beanies–that

all can come with intricate patterns. The complexity of these patterns is hard to achieve in real life, as in the virtual world. Maya does not have a plug-in to create models of knitted clothing.

1.3. Design Goals

1.3.1 Target Audience.

The target audience for our tool is mainly artists who deal with 3D modeling. Since the tool will take a polygonal mesh of a character's garment as input, it is assumed that an artist will be the most likely person to use our tool to transform the polygon mesh into a knitted mesh.

1.3.2 User Goals and Objectives

The user will be able to quickly and easily create a knitted yarn-level model of a given polygon mesh. For the given mesh input, the user will be able to control the overall knitting pattern of the garment by specifying desired stitch types and time permitting, desired knitting direction. Given the user-specified parameters, the tool will quickly and easily generate a yarn-level model of the user's input mesh.

1.3.3 Tool Features and Functionality

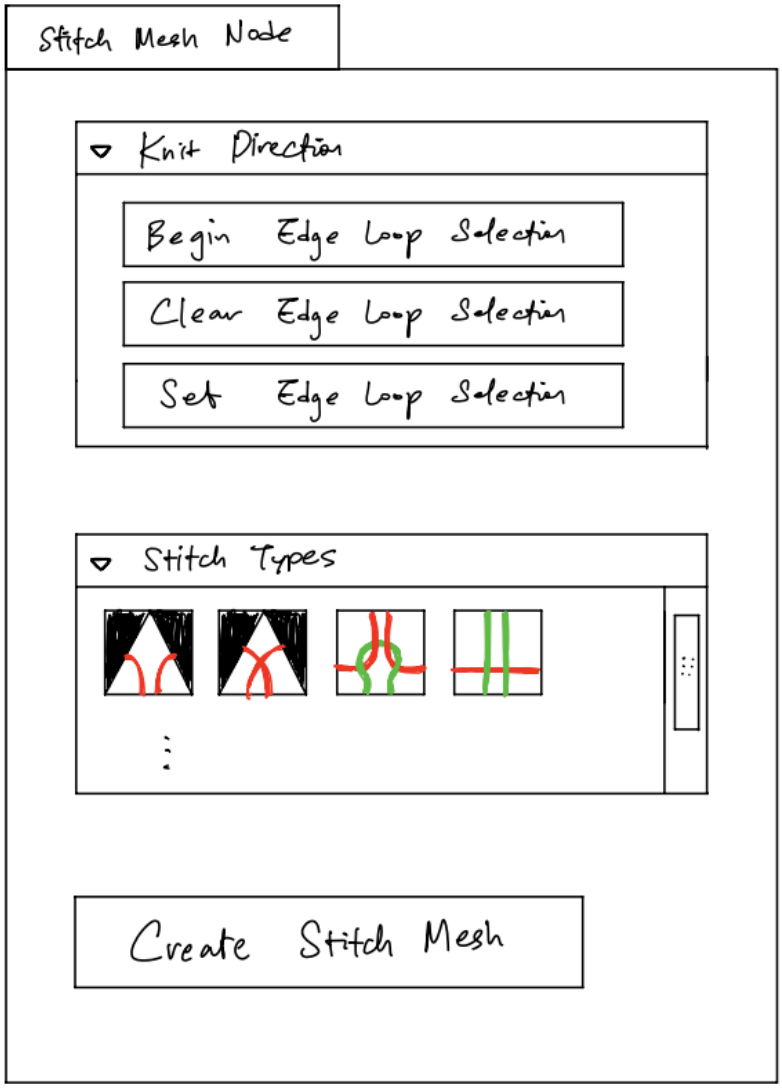
The user will be able to change the parameters and stitch types that go into creating the polygon mesh. Given a polygonal mesh that defines the garment's surface, the user will be able to label the components of the mesh to specify the knitting direction over the surface. In real-world terms, this denotes where one would start knitting this garment and in which direction the knitter would continue knitting until the garment is completed. Using these knitting directions and the parameters specified by the user, the tool will create a high-resolution stitch mesh. The stitch mesh serves as an abstract representation of a yarn-level cloth model. Each face of the stitch mesh corresponds to a particular stitch of the yarn-level model. The stitch type of a face can be assigned by the user from a collection of predefined stitch types.

1.3.4 Tool Input and Output

Input: Polygonal Mesh

Output: Detailed yarn-level model of the given mesh

1.4. User Interface



1.4.1 GUI Components and Layout

The GUI for this tool is relatively simple, consisting of only one Stitch Mesh Attribute Editor window containing the controls for the input to our tool. The user can access this GUI by creating a Stitch Mesh Node from the Maya Menu after selecting a low-poly mesh object representing the garment.

The Knit Direction window allows the user to select consecutive edge loops in order to specify where the knitting direction is oriented. If the user changes their mind, the user can also clear the selection and start over. After specifying the knitting direction for the entire surface of the mesh, the user can set the edge loop direction and move on to assigning specific stitch types to specific faces of the mesh.

The Stitch Types window provides a selection of Stitch Types to choose from. The user is able to select a face of the mesh using Face Mode and assign one of the stitch types to that face.

Once the user is satisfied with the above parameters, the user can click the "Create Stitch Mesh" button in order to create the Stitch Mesh version of the initial input mesh.

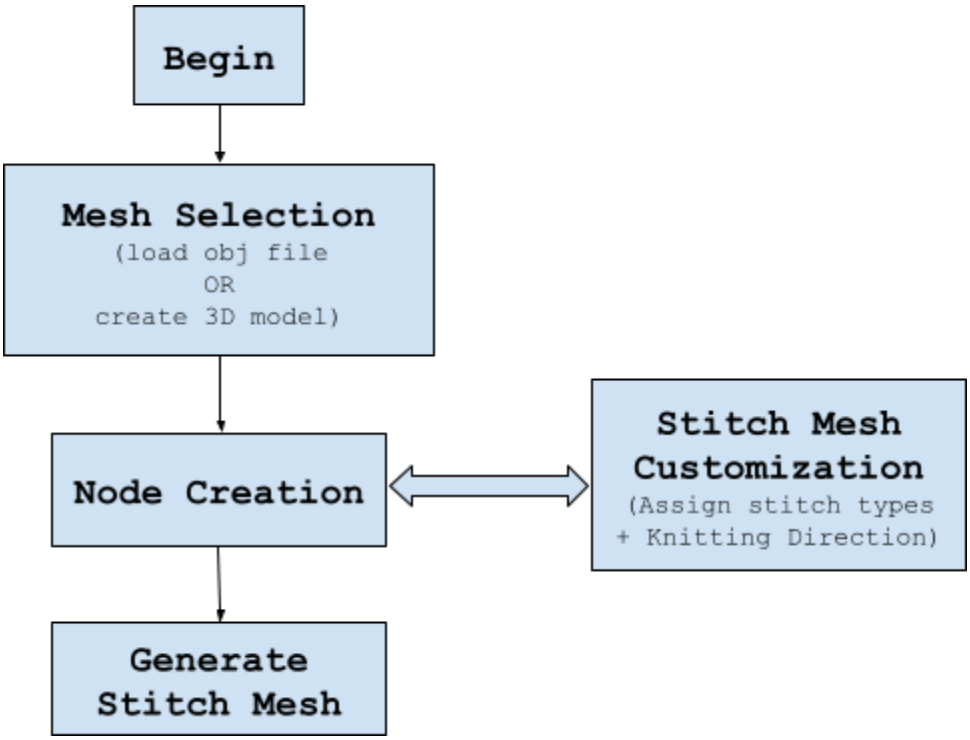
1.4.2 User Tasks

Once a user creates the Node, all that is required of the user is to specify the knitting direction across the entire surface of the mesh by selecting consecutive edge loops. The user will first select a boundary edge loop to specify where the knitting will begin. Consecutive edge loops will indicate where the knitting direction is oriented.

After specifying the knitting direction across the entire mesh, the user can set the selection in order to start assigning stitch types to individual faces. If the user wants to assign a special stitch type to a particular face, the user can go into Face Mode and select a face from the mesh. While the face is selected, the user can click on a stitch type from the selection in order to assign the stitch type to the face.

The user will not need to have any technical background knowledge in order to use the tool. The user will only need to know what stitch types they would like to see in the garment as well as the knitting direction that will ultimately affect the overall look, but this can be experimented with and is left up to the user's creativity.

1.4.3 Work Flow



1. Load the obj file or create a low-poly mesh of the garment - REQUIRED INPUT
2. Create Stitch Mesh Node
3. Select Knitting Direction over the surface of the mesh by selecting consecutive Edge Loops
4. Assign Stitch Types to Individual Faces (if desired)
5. Generate the Stitch Mesh - OUTPUT

2. AUTHORING TOOL DEVELOPMENT

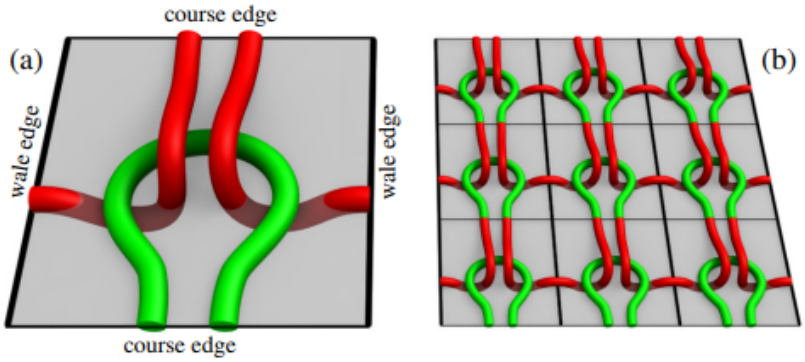
2.1. Technical Approach

2.1.1 Algorithm Details

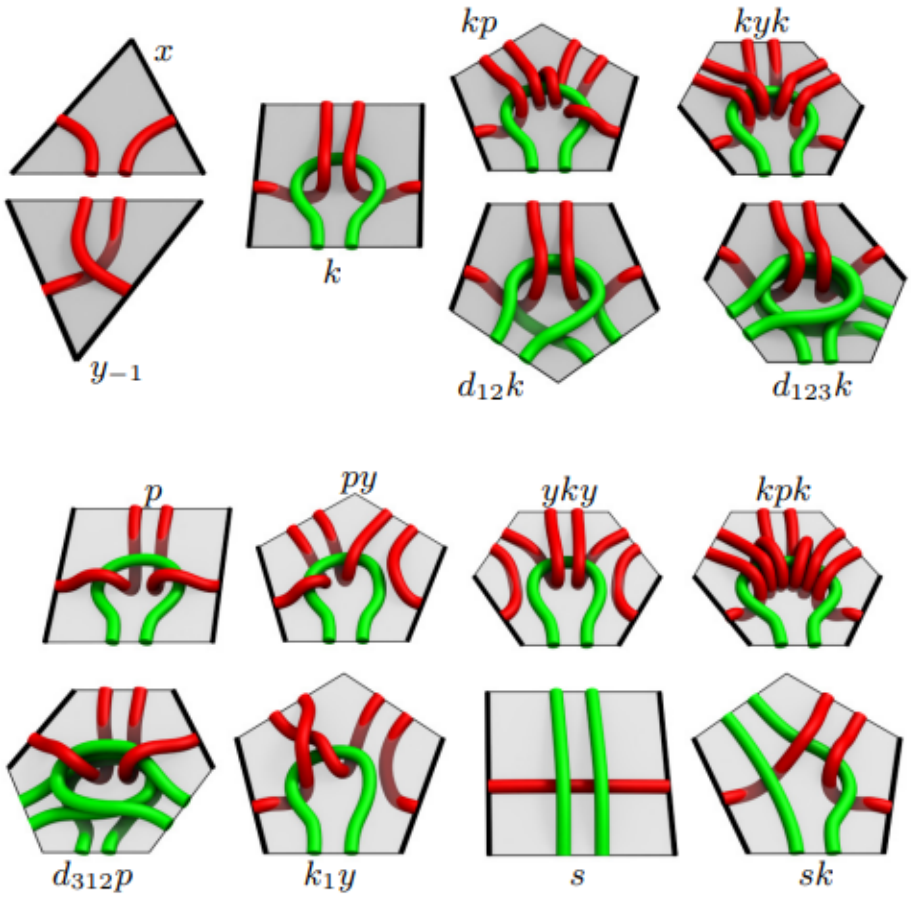
Our interactive modeling algorithm consists of the following components:

Stitch Types: fundamental actions for generating valid knitting stitches; can be combined to form more complex stitches

- *k*: Knit (pull yarn through an existing loop from the back side towards the front)
- *p*: Purl (pull yarn through an existing loop from the front side towards the back)
- *y*: Yarn-over (wrap yarn around knitting needle without pulling it through a loop)



<u>Pre-defined Stitch Type Configuration Examples:</u>



For Knitting non-planar shapes & Various patterns:

<u>Increases:</u> formed by using combinations of *k, p, y* on one existing loop

- used for increasing # of stitches on the next row

<u>Decreases:</u> Stitch mesh faaces with multiple bottom edges decrease the number of stitches on the current row as compared to the previous row

Possible stitch types that can be generated:

# bottom edges	**# top edges**	**possible stitches**
1	1	*k, p*
1	2	*kp, ky, yk, py, yp*
1	3	*kpk, kyk, yky, kpy, ykp, pkp, pyp, you, pky, ypk*
2	1	$d_{12}k$, $d_{21}k$, $d_{12}p$, $d_{21}p$
3	1	* $d_{123}k$, $d_{132}k$, $d_{213}k$, $d_{231}k$, $d_{312}k$, $d_{321}k$, $d_{123}p$, $d_{132}p$, $d_{213}p$, $d_{231}p$, $d_{312}p$, $d_{321}p$

***** when forming a decrease stitch, the order in which the existing loops are stacked together determines the direction that the stitch tends to twist; represent decreases with the symbol *d* followed by a subscript that determines stacking order

PSEUDO CODE:

We want to determine default configuration of stitch types for each face in the mesh.

Given a low-poly input mesh and the user-specified knitting directions over the mesh surface:

- Separate poly mesh into rows of faces:
    - Handle borders of the cloth: Check if row of faces contains Boundary Edge Loop (specified in the knitting direction step by the user)
        - If this row is the first row:
            - set the faces in this row to be *cast-on* stitches
        - If this row is the last row:
            - set the faces in this row to be *bind-off* stitches
- For each face in the row:
    - Check neighboring faces
        - If I have neighbors (not a cast-on or bind-off):
            - Depending on # of course & wale edges, assign default stitch type to this face from list of pre-defined stitch types
- Return Stitch Mesh

Assumptions & Simplifications:

While the paper implements offline relaxation, a yarn-level physical simulation of the knitted garment, we will not include this functionality in Knit-It based on a realistic scope for this assignment.

2.1.2 Maya or Houdini Interface and Integration

For this project, all UI components such as dialogs and user settings will be implemented in MEL. Users will be able to select either a 3D model that they create in Maya or load an obj file to use as input.

The actual running of the algorithm and assignment of the stitch meshes will be done within the C++ API. It will receive mesh data and other inputs from the MEL script and perform the necessary iterations of the algorithm, automatically generating a knitted version of the mesh. More specifically, the API will create a new mesh into a new Maya object.

We will make use of the Maya MfnMesh class and its functions to handle the garment input and outputs and access attributes within the mesh. We also envision using the MObjectArray and MSelectionList classes in order to keep track of face loops (for knitting direction) and assigning stitch types to faces based on their neighbors.

2.1.3 Software Design and Development

StitchFace

The building block which represents a portion of the knitted garment; a cell where two yarns interact. Consists of the top part of one yarn loop, and the bottom part of another yarn loop that is pulled through forming a stitch. These are individual units that are independent since they connect only at single points between cells, and can create complicated structures by composing other StitchFaces together. StitchFaces do not necessarily have to be quads, but each face must have exactly two wale edges that connect it to adjacent stitches on the same row. A face can have any number of top and bottom course edges.

CourseEdge

Edges of the StitchFace that are aligned with the course direction; each course edge has two yarns crossing it

WaleEdge

Edges of the StitchFace that are aligned with the wale direction; each wale edge has a single yarn crossing it. Two wale edges separate the course edges into two groups: *top* and *bottom*, such that the wale direction on the face points from the bottom edges towards the top edges.

StitchType

Pre-defined stitch mesh types as defined by the paper. We will build our own library containing these stitch types, and allow the user to select stitch types to assign to individual faces on the mesh.

Third-Party Software

- tinyobj: obj file loader (maybe; might be able to get away with Maya's default import functionality)

2.2. Target Platforms

2.2.1 Hardware

- 64-bit Intel® or AMD® multi-core processor with SSE4.2 instruction set; Apple Mac models with M series chip are supported under Rosetta 2 mode
- 8 GB of RAM (16 GB or more recommended)
- Maya Certified Hardware for Maya2022

2.2.2 Software

- Maya2022
- Microsoft® Windows® 10 or higher; Apple® macOS® 11.x, 10.15.x, 10.14.x, 10.13.x operating system

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

The Knit-It tool will contain a basic Node UI that will allow users to select a polygonal mesh,

2.3.2 Beta Version Features

User will be able to

2.3.3 Description of any demos or tutorials

3. WORK PLAN

3.1. Tasks and Subtasks

We divided the development of our tool into the following tasks:

Task 1 – Create Node for Stitch Mesh Tool (Krystal)

Duration: 2 days

Implement the GUI for the Node required for the tool in MEL. Design the main window, the obj file loader (if we end up using tinyobj), and make sure the Node is accessible from the Maya Menu. Create the foundation for all of the functionality to follow.

- ***Subtask 1.1. Basic MEL Script***

  Create the base MEL script that populates the tool in the Maya Menu. Borrow code from previous Maya HW assignments.

- ***Subtask 1.2. Stitch Type Library Dropdown***

  Create an empty dropdown window where the stitch type library and the icons of all stitch types will go.

- ***Subtask 1.3. Knitting Direction Window***

  Create an empty window where the GUI elements for the knitting functionality will go. Make buttons for begin, set, and clear for the Edge Loops.

Task 2 – Build Library of 3D Models of Stitch Types (Monique)

Duration: 4 days

Create a library of pre-defined stitch types. Create icon images that will be selectable in the Node. Create 3D models of the 15 different stitch types from the paper.

- ***Subtask 2.1. Create Base Spline Models***

  Think of Stitch Mesh Types as splines. Model the shape of different stitch types using the spline tool. Extrude in order to create 3D geometry for the yarns.

- ***Subtask 2.2. Compile Library***

  Once all stitch types are built, organize them into a single folder accessible to Maya. The library should be visible in the Node attribute editor under a drop down and contain all icons that match the stitch type.

- ***Subtask 2.3. Tweaking & Testing***

Manually test the stitch types by piecing them together. Start with an easy surface mesh, like a subdivided cube. Tweak the thickness and shape of the stitches as needed.

Task 3 – Knitting Direction Functionality (Monique/Krystal)

***NOT REQUIRED FOR TOOL TO WORK - "DESIRED" FUNCTIONALITY**

Duration: 2.5 weeks

Implement functionality to allow the user to specify the knitting direction across the entire surface of the mesh by selecting consecutive edge loops. The user can select a boundary edge loop to specify where the knitting will begin. Consecutive edge loops will indicate where the knitting direction is oriented. This module will be implemented using MEL to access the selected attributes of the mesh.

- ***Subtask 3.1. Implement Default Knitting Direction***

  Implement functionality to assign knitting direction to the faces of the mesh automatically. Assume that all meshes are created with a single direction (top-to-bottom).

- ***Subtask 3.2. "Begin Edge Loop Selection" Button***

  Implement functionality to allow the user to be able to click edge loops on the borders of the cloth. Selecting consecutive edges will register the knitting direction for that particular row of faces.

- ***Subtask 3.3. "Set Edge Loop Selection" Button***

  Develop functionality to finalize the edge and face loop selections. Clicking on the button should save the knitting direction configurations so that the user will be able to assign stitch types to the faces of the mesh.

- ***Subtask 3.4. "Clear Edge Loop Selection" Button***

  Implement functionality to clear all the selections. Time permitting, break this functionality up into an "undo single loop" so that the user will not have to start all over again.

Task 4 – Stitch Type Assignment Algorithm (Monique/Krystal)

Duration: 2.5 weeks

Implement functionality to allow the user to specify the knitting direction across the entire surface of the mesh by selecting consecutive edge loops. The user can select a boundary edge loop to specify where the knitting will begin. Consecutive edge loops will indicate where the knitting direction is oriented. This module will be implemented using MEL to access the selected attributes of the mesh.

- ***Subtask 4.1. Create Test-app for Default Algorithm***

  Since there is no algorithm provided in the paper, using valid stitch configurations described in the paper, come up with an algorithm that assigns default stitch configurations given a low-poly mesh. For now, use the most simple stitch types.

- ***Subtask 4.2. Stitch Type Assignment in Maya (default assignment)***

  Once we confirm the test app is working, implement the main default assignment algorithm in C++ to use in Maya. Verify that the test-app functionality is working in Maya. Test out with a low-poly mesh and use the most simple stitch types.

- ***Subtask 4.3. Allow Stitch Type Editing (*Time Permitting*)***

  Implement functionality so that the user can edit individual faces by assigning other stitch types after the initial assignment.

Task 5 – Testing/Debugging (Monique/Krystal)

Duration: 3 weeks

The final task will be general testing and debugging of the Maya plug-in, as well as finalizing all necessary documentation and creating any Demo materials.

- ***Subtask 5.1. Test with First-Time Users***

  Ask other people to test out our tool. Provide a brief explanation of the tool, but let the users attempt to use the tool by themselves. Observe and note any difficulties in using the tool.

- ***Subtask 5.2. Create Demo Materials***

  Download an obj or create a low-poly mesh of a garment we want to use for the demo. Verify that the plug-in works for these chosen garments (preferably sock and sweater).

- ***Subtask 5.3. Prepare Final Presentation***

  Depending on the final presentation instructions, create a Google Slide that gives an overview of our authoring tool. Create a video presentation showcasing the functionality of the plug-in.

3.2. Milestones

3.2.1 Alpha Version

- Create Node for Stitch Mesh Tool (Tasks
- Stitch Type Assignment Algorithm (Default Version)
- Knitting Direction Assignment (Default Version)
- Basic Demo complete

3.2.2 Beta Version

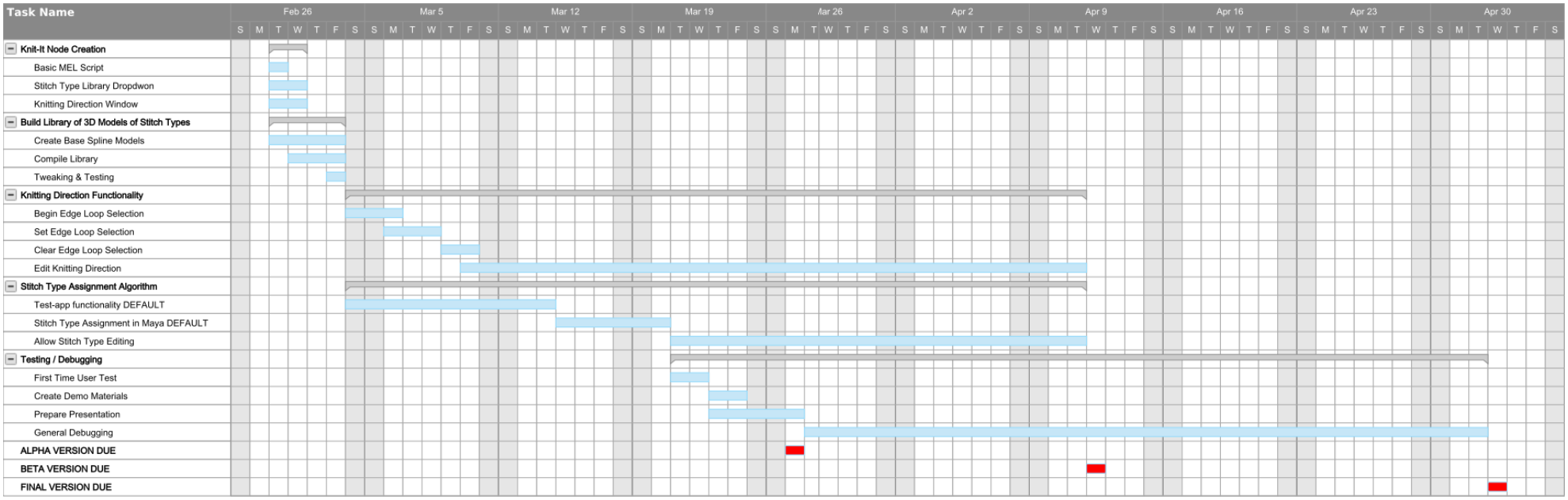
- User Testing complete
- Editable Stitch Type Assignment
- Editable Knitting Direction Assignment

3.2.2 Final Version

- Final Presentation / Demos
- Last minute Debugging

3.3. Schedule

Gantt Chart

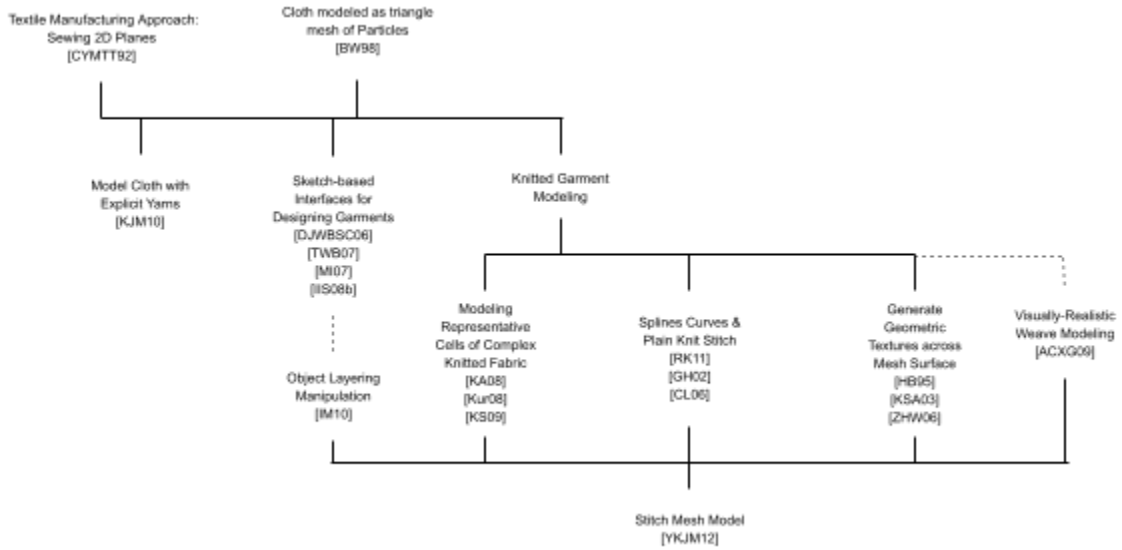


4. RELATED RESEARCH

Our authoring tool “Knit-It” is based on the modeling technique described in the “Stitch Meshes for Modeling Knitted Clothing with Yarn-level Detail” paper by Yuksel et. al. The approach builds yarn-level models of complex knitted garments from polygonal models that represent a large-scale surface of knitted cloth. The output is a finer mesh representing the layout of the stitches in the garment.

In this literature survey, we trace from the seminal works **[CYMTT92]**, **[BW98]**, starting with modeling and simulating clothing with a triangular mesh of particles, to the modeling technique of sewing two individual 2D panels together, and finally modeling knitted clothing with a stitch mesh from a polygonal model.

Research Evolution Tree



Contents

In the seminal paper **[BW98]**, Baraff and Witkin model and simulate cloth using a triangular mesh of particles. This work is related to **[CYMTT92]**, where clothing is modeled using two individual 2D panels seamed together, similar to actual textile manufacturing processes. The resulting garments can be worn by and attached to synthetic actors. While our authoring tool does not consider animating the cloth, the resulting garments worn by the synthetic actors can be animated using internal elastic forces and external forces of gravity, wind, and collision detection, similar to the **[BW98]** paper. Other approaches like the one described in **[KJM10]** have looked at modeling cloth with explicit yarns (each yarn is modeled as an inextensible B-spline tube).

More approaches to the geometric modeling of garments include sketch-based interfaces for designing garments and other 3D objects composed of 2D pieces. The geometric garment modeling method used in **[TWB07]** is borrowed from the method used in **[DJWBSC06]**, which generates an initial 3D surface from a sketch-based user input using a distance field technique. In addition, **[TWB07]** allows the user to specify additional seam lines on the garment by sketching them on the surface. **[MI07]** expands the idea into creating 2D patterns and accompanying 3D models for plush toys from an initial sketch, with a particular focus on ensuring that the appearance of the 3D model generated from the sketch is accurate to the final output of sewing the 2D pattern together in real life. This then eventually evolves into **[IIS08b]**, narrowing their scope to creating knitted animals, which allows users to visualize their sketches as a 3D knitted model and provides users with patterns to knit the animal themselves. While it should be noted that Igarashi et al.'s representation of knitted objects consists of knit textures projected onto surfaces, due to the nature of the project, greater care is taken to align the grain in a more realistic manner, and the 2D patterns show the proper knitting pattern necessary to create the final product.

For yarn-based garments, researchers have looked at replicating the manufacturing processes used by knitting machines. In **[DB06]**, cloth is modeled with filaments, which are represented by a series of rigidly connected beam elements. These processes produce correct knitted patterns, but are limited to small patches of material and are only able to support a limited set of stitch types much like real knitting machines. **[ME98]** uses input data from a knitting machine to help generate simplified knitted topologies. **[KJM11]** expands the idea even further, implementing a semi-automated process involving models of individual knit loops that can be tiled together to form cloth; however, constructing a new garment requires the user to write code specifying how loops are laid down, to create geometric models for any new types of loops used, and to verify that the final result is topologically correct.

There has also been work in the textile community on the geometric modeling of knitted materials. Other works generate knit geometry using spline curves and focus on the plain knit stitch **[RK11]**. **[GH02]** uses a 3D cylindrical and uniform solid yarn model, such that its central axis is a space curve, using non-uniform rational B-spline (NURBS) surfaces to achieve a true 3D solid representation of warp knit loops and structures. This method assumed that the shape of the yarn after knitting was curved and the loops were allowed to rotate such that the geometry of each loop was no longer symmetrical. **[CL06]** builds on the method by allowing predictions of both fabric dimensions and wale spirality, showing improved agreement for both course spacing and spirality in the fabric. Other research has looked at modeling representative cells of complex knitted fabric **[KA08]**, **[Kur09]**, **[KS09]**, but depending on the pattern, these cells might have to contain many individual stitches, and combining multiple patterns in a single fabric requires the boundaries of each representative cell to be carefully and correctly matched up.

Due to the similarity between how textures and knitted yarn are represented visually, as touched upon in **[HB95]** and **[KSE03]**, texture synthesis-like methods for generating

geometric textures across a mesh's surface have been explored in papers such as **[ZHW06]**, which allows the user to input a mesh and a desired geometric pattern sample, and then receive an output of the pattern stitched together and deformed over the mesh's arbitrary surfaces. While this method may seem to be intuitively applicable to applying a knit geometry sample to the model of a clothing item for the desired knit look, such techniques do not take into account the intricacies of real-life knit fabrics, whose threads must be joined and overlapped in a particular fashion order to prevent unraveling, and are thus insufficient for representing physically realistic knit models.

Most recently, before the publication of our authoring tool's paper, **[ACXG09]** was able to transform any given mesh into a plain-woven object that is visually realistic, and **[IM10]** developed a way for users to easily manipulate the layer order of overlapping 3D objects (even self-layering). Both of these papers lead nicely into the development of Yuksel et al.'s paper **[YKJM12]**, due to the similarities in both construction and appearance between weaving and knitting, as well as the inherent abundance of self-overlapping of yarn strands and the need to be able to quickly modify these overlapping areas during yarn cloth generation.

References

[CYMTT92] CARIGNAN, M., YANG, Y., THALMANN, N. M., AND THALMANN, D. 1992. Dressing animated synthetic actors with complex deformable clothes. *ACM SIGGRAPH'92*, 99–104.

[BW98] BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. *ACM SIGGRAPH'98*, 43–54.

[KJM10] KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM T. Graph. (SIGGRAPH'10) 29*, 4, 105.

[TWB07] TURQUIN, E., WITHER, J., BOISSIEUX, L., CANI, M.-P., AND HUGHES, J. 2007. A sketch-based interface for clothing virtual characters. *IEEE Comp. Graph. and Applications 27*, 1, 72–81.

[DJWBSC06] DECAUDIN, P., JULIUS, D., WITHER, J., BOISSIEUX, L., SHEFFER, A., AND CANI, M.-P. 2006. Virtual garments: A fully geometric approach for clothing design. *CG Forum (Eurographics) 25*, 3, 625–634.

[MI07] MORI, Y., AND IGARASHI, T. 2007. Plushie: an interactive design system for plush toys. *ACM T. Graph (SIGGRAPH'07) 26,* 3, 45.

[IIS08b] IGARASHI, Y., IGARASHI, T., AND SUZUKI, H. 2008. Knitting a 3D Model. *CG Forum (Eurographics) 27*, 7, 1737–1743.

[DB06]DUHOVIC, M., AND BHATTACHARYYA, D. 2006. Simulating the deformation mechanisms of knitted fabric composites. *Composites Part A: Applied Science and Manufactur*. 37, 11, 1897–1915.

[ME98] MEISSNER, M., AND EBERHARDT, B. 1998. The art of knitted fabrics, realistic physically based modelling of knitted patterns. *CG Forum (Eurographics) 17*, 3, 355–362.

[KJM11] KALDOR, J. 2011. *Simulating Yarn-Based Cloth*. Ph.D. thesis, Cornell University.

[RK11] RENKENS, W., AND KYOSEV, Y. 2011. Geometry modelling of warp knitted fabrics with 3D form. *Textile Research Jour. 81*, 4, 437–443.

[GH02] GOKTEPE ¨ , O., AND HARLOCK, S. C. 2002. Three-dimensional computer modeling of warp knitted structures. *Textile Research Jour*. *72*, 266–272.

[CL06] CHOI, K., AND LO, T. 2006. The shape and dimensions of plain knitted fabric: A fabric mechanical model. *Textile Research Jour. 76*, 10, 777–786.

[KA08] KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM T. Graph. (SIGGRAPH'08) 27*, 3, 65.

[Kur09] KURBAK, A. 2009. Geometrical models for balanced rib knitted fabrics part I: Conventionally knitted 1x1 rib fabrics. *Textile Research Jour. 79*, 5, 418–435.

[KS09] KURBAK, A., AND SOYDAN, A. S. 2009. Geometrical models for balanced rib knitted fabrics part III: 2x2, 3x3, 4x4, and 5x5 rib fabrics. *Textile Research Jour. 79*, 7, 618–625.

[HB95] HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *ACM SIGGRAPH'95*, 229–238.

[KSE03] KWATRA, V., SCHO ̈DL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM T. Graph. (SIGGRAPH'03) 22*, 3, 277–286.

[ZHW06] ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM T. Graph. (SIGGRAPH'06) 25*, 3, 690.

[ACXG09] AKLEMAN, E., CHEN, J., XING, Q., AND GROSS, J. L. 2009. Cyclic plain-weaving on polygonal mesh surfaces with graph rotation systems. *ACM T. Graph. (SIGGRAPH'09) 28*, 3, 78.

[IM10] IGARASHI, T., AND MITANI, J. 2010. Apparent layer operations for the manipulation of deformable objects. *ACM T. Graph. (SIGGRAPH'10) 29*, 4, 110.

[YKJM12] Yuksel, C., Kaldor, J. M., James, D. L., & Marschner, S. (2012). Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Transactions on Graphics (TOG), 31(4)*, 1-12.

5. FINAL REPORT

5.1. Results

5.1.1 Documentation/Tutorial

In order to use our authoring tool, users should follow these steps:

1. Download and unzip the plugin package.
2. Drag the folder's contents (not the folder itself) to their computer's
3. Maya plugin folder.
4. Open Maya 2022, and either start a new project or open an existing project.
5. From Maya's menu bar, click on Windows > Settings/Preferences > Plug-In Manager (Fig. 5.1a).
6. In the Plug-In Manager window, there should be a collapsible section corresponding to the user's local Maya plugin folder. Expand the section.
7. Click on the Loaded checkbox next to knitInstanceNode.py. A new menu item called KnitInstance should appear in Maya's menu bar (Fig. 5.1b).
    a. Note: There will be a SystemError that appears in Maya's terminal (usually the bottom-right corner of the application window), but it can be ignored, as it does not affect any functionality.
8. If the user wants to test the default settings:
    a. Under the KnitInstance menu (Fig. 5.1c), click on Create Default Stitch Plane to view the main stitch pattern on a flat surface.
    b. Click on Create Default Stitch Cylinder to view the main stitch pattern on a curved surface.
9. If the user would like to create a knitted version of their own custom mesh:
    a. Create a custom mesh in the current project window, or import an .obj or .mb file.
    b. Select the mesh in Object Mode.
    c. Under the KnitInstance menu, click on Create Stitch Mesh with Selected Geometry.
    d. In the pop-up window that shows up (Fig. 5.1d), the user may elect to keep the original mesh underneath the knit pattern, select which stitch-face to use (the default stitch from the Yuksel et al. paper or our custom cross-hatch stitch), and

subdivide their original mesh up to 3 times to achieve the desired level of detail.

e. Click the “Knit It!” button to create the stitch mesh. Depending on how detailed the original mesh is and how many subdivisions are desired, this may take up to a few minutes.

5.1.2 Content Creation

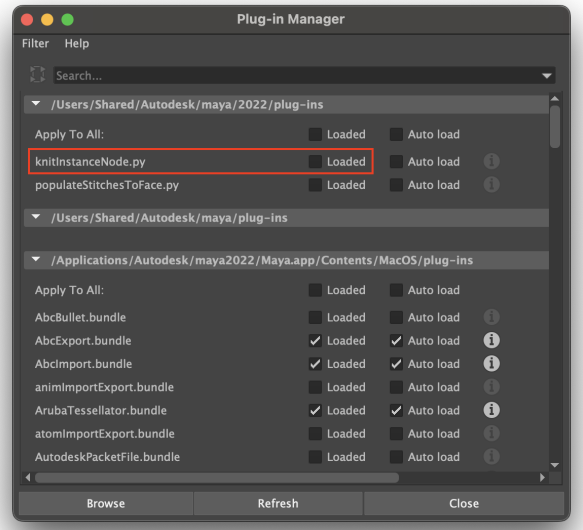


Figure 5.1a: Maya’s Plug-in Manager window (MacOS). The plug-in to be loaded, knitInstanceNode.py, is highlighted with a red rectangle.

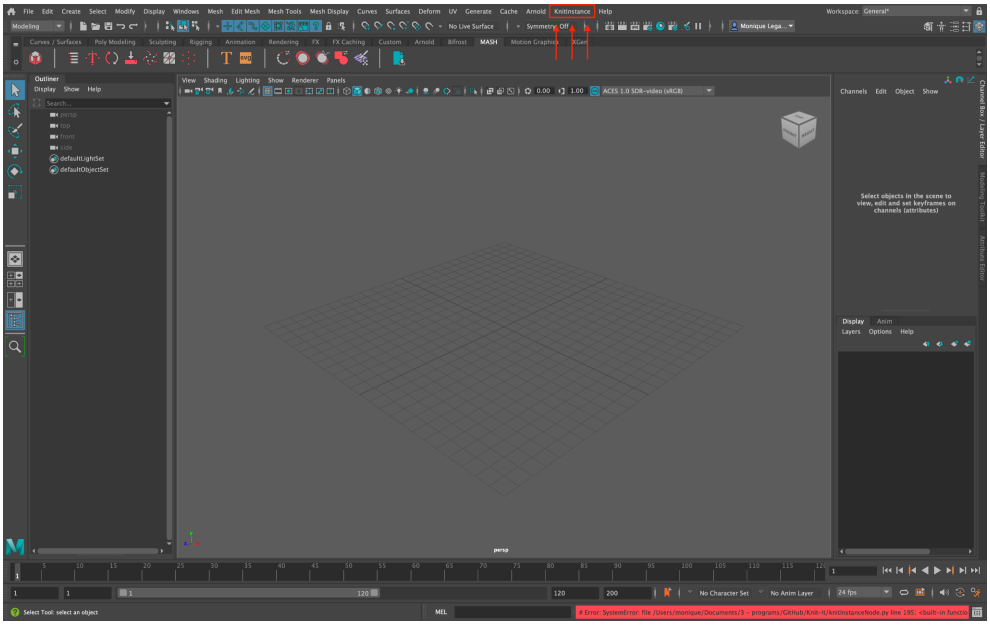


Figure 5.1b: The KnitInstance menu (highlighted with red rectangle and arrows) as it appears in the Maya menu. Note the error message in the bottom-right, which can be ignored.

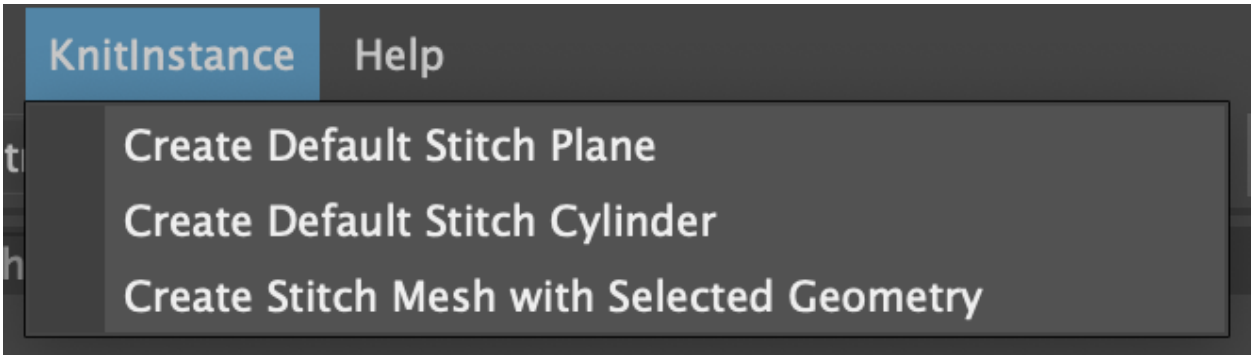


Figure 3: The dropdown selections in the KnitInstance menu.

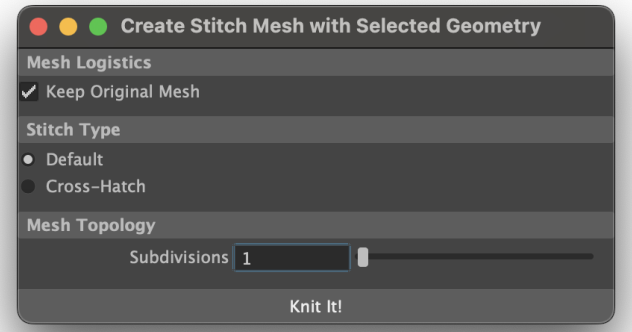


Figure 4: The pop-up window for creating custom stitch meshes.

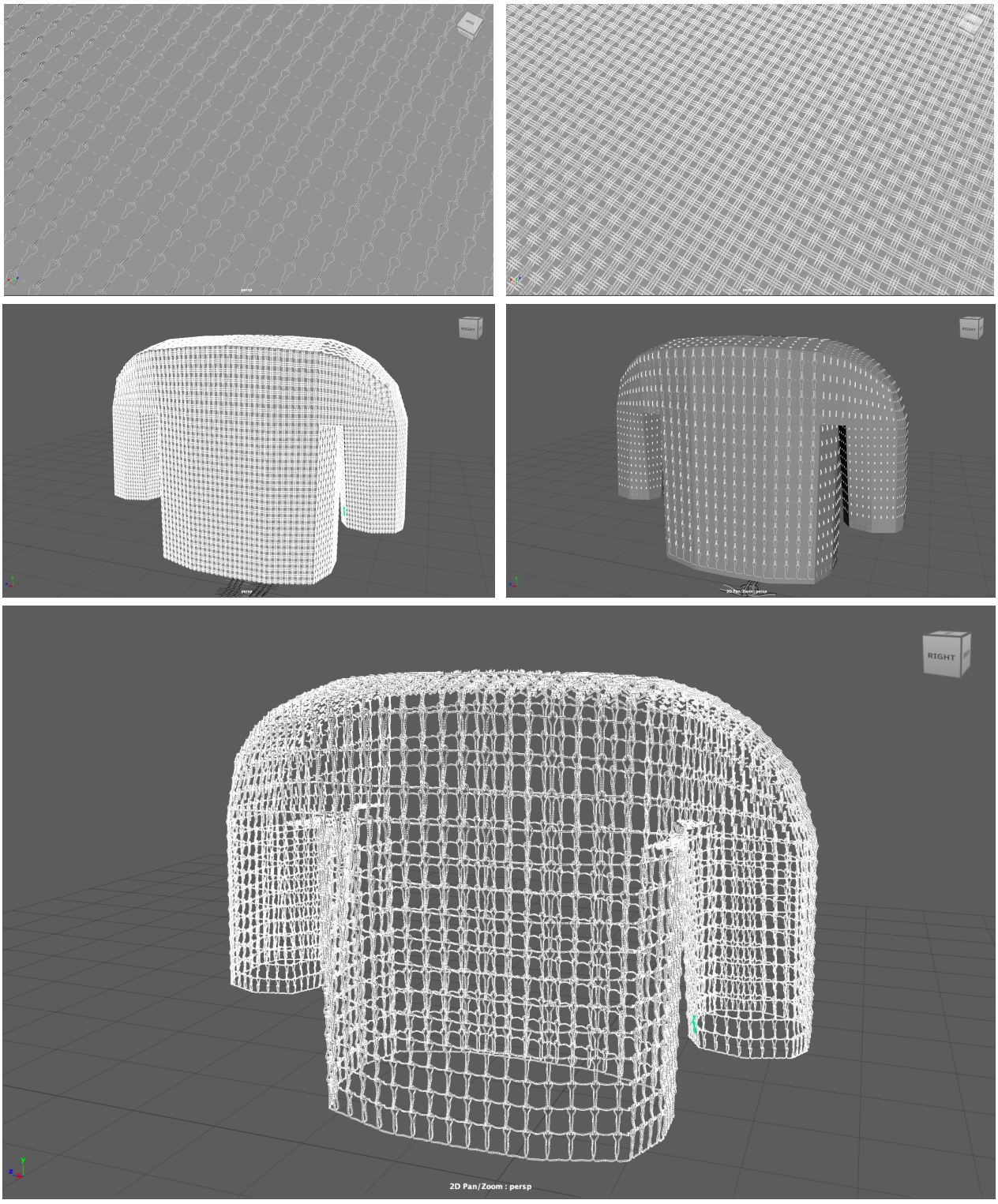


Figure 5.1e: Some examples of what can be rendered using our authoring tool.

5.2. Discussion

5.2.1 Accomplishments

Throughout the process of working on our authoring tool, we believe we have accomplished much of what we set out to do. We were able to:

- Create a library of stitches to be used in our authoring tool, including a custom cross-hatch stitch that was not present in the original paper.
- Create a Maya plugin (albeit Python-based, rather than the MEL-based one that we had planned to do) that can be easily launched from the Plug-In Window of Maya.
- Populate a given quad-based mesh with stitches along its faces.
- Recognize when a mesh face is an edge case and populate that face with a cast-on stitch, rather than the main stitch-face.
- Warp the stitch-face to approximately match the shape of an irregular quad face.
- Allow the user to adjust parameters to their liking, including choosing whether to keep the original mesh underneath the stitch mesh, selecting which stitch-face to use, and deciding how many subdivisions to apply to the mesh for their desired level of detail.

Our journey was not without problems. Some challenges we encountered were:

- Covering the entire spectrum of edge cases on a stitch mesh. There are special stitches that comprise the edges of a knitted object, and although we did create the stitch-faces and add them to our library, due to the nature of how MEL stores and allows users to find data on faces, we could identify *whether* a face was an edge case, but not *which* edge case. For example, if a face had less than 4 adjacent faces (sharing the same edge), we knew it needed something other than the main stitch-face. However, for the majority of our development process, we could not identify the relative positions of these adjacent faces, making it impossible to tell if the face was, say, a right-edge or a top-left-corner. As a placeholder, we have populated the face of each edge case with a cast-on stitch, which works well enough for our purposes.
- Accurately warping the stitch-face to match the shape of an irregular quad face. We had originally planned to loop through the vertices of the stitch and LERP them according to the positions of the quad face's vertex corners. However, we were not able to identify a way to find the positions of the vertices of the quad until about a week before the final

project was due. Then, LERPing the vertices of the stitch was too complex and computationally expensive. Our TA, Sebastian Vargas, suggested approximating the warp by scaling the stitch by the quad face's average X and Z side lengths. (We now refer to this as the Vargas Approximation.)

Unfortunately, due to time constraints and technical issues, we were not able to implement the following features:

- User-specified knitting direction. For simplicity, we assume that the knitting direction is in the positive Y-direction.

5.2.2 Design Changes

Design-wise, our implementation stayed mostly true to what we outlined in this design doc. However, there were some coding problems with Python-based solutions, which did not have MEL equivalents, so we spent the last weekend translating all of our code into Python. The functionality remains the same.

5.2.3 Total Man-Hours Worked

Our total work, time-wise, amounts to about 163 hours; Krystal worked approximately 78 hours, and Monique worked approximately 85 hours.

5.2.4 Future Work

In order to develop Knit-It, in its current state, into a fully functional tool usable by an artist, here are some features that we believe could be added or improved:

- Full stitch edge-case coverage. Theoretically, we have a solution for this (involving the newfound knowledge of how to retrieve a mesh component's position relative to another component, thanks to our work with stitch warping), but it required more time and thought than we were able to spare at the end of our development process.
- The ability to specify and change knitting direction.
- Optimizations for the stitch population process, so that it can run faster. Currently, our low-poly sweater model takes about 2 minutes to transform into a knit mesh.
- More stitch types to allow for greater customization.
- A cleaner UI for the Custom Stitch Mesh pop-up menu. We added basic functionality in a way that would be easy for the user to

understand and navigate, but it would be better if, for example, the stitch type selection included pictures of the different stitches.

5.2.4 Third-Party Software

Our plug-in does not make use of any third-party software. All code has either been written by us or properly accredited to answers found on online forums via comments.

5.3. Lessons Learned

Throughout the Knit-It development process, we learned many things that we wish we had known in advance. Here is a non-exhaustive list of what we discovered:

- OpenMaya, a library for Python, has MVector, MMatrix, and MPoint classes that make mesh manipulation much easier than having to deal with the built-in MEL data structures.
- When we tried to query mesh components, Maya would default to giving us slices of the components' indices if they were numerically consecutive – for example, "pCube1.f[1:3]" instead of "pCube1.f[1] pCube1.f[2] pCube1.f[3]". We spent weeks trying to find ways around this via string-splitting, but ended up finding out MEL has a built-in function for this called filterExpand.
- To get the position of a component A relative to another component B, first get the normal and world position of B and use these to get the transformation matrix of B. Then get the world position of A and *post-multiply* the inverse of the transformation matrix. (This is basic frame-of-reference mathematics, but it took a while for us to be able to translate it into MEL/Python.)
- To get the absolute file path of a Python file, you first need to import os, and then call "filepath = os.path.dirname(os.path.abspath(__file__))" . This was essential to being able to import stitches without requiring the user to hardcode the file path, which we realized might be difficult for artists who have less familiarity with such things.

5.4. Acknowledgments

Thank you to our professor, Dr. Stephen H. Lane, for his guidance and feedback; our course TAs, especially Sebastian Vargas, who supported us with his own knowledge on stitching and mesh manipulation; and to our fellow classmates, who all continue to inspire us with their infinite passion for, and steadfast dedication to, the world of computer graphics.