CIS6600: Advanced Topics in Computer Graphics and Animation

Wood Knot Editor

Maya Plug-in Design Document

Wenqing Wang Xu Zhang

Based on:

Procedural Texturing of Solid Wood with Knots

Maria Larsson, Takashi Ijiri, Hironori Yoshida, Johannes A. J. Huber, Magnus Fredriksson, Olof Broman, and Takeo Igarashi

PROJECT SUMMARY

Our project aims to create a Maya plug-in that uses procedural methods to generate realistic wood textures with knots. We anticipate that our tool will streamline artists' modeling workflows by reducing the time spent manually drawing or adjusting textures.

To achieve this, our tool builds upon previous work in procedural texturing. Although wood texturing has been studied extensively, there have been limited attempts to model nodes within wood textures. Knots are distinctive areas where wood grain grows in a circular pattern, resulting in unique visual characteristics. Our goal is to address the challenge of modeling wood with knots, allowing for the automatic generation of more realistic wood textures.

Our tool will take the skeletal structure of a tree log as input and generate a three-dimensional scalar field that represents the added growth time, defining the annual ring pattern of the volume. Initially, separate fields are calculated around each strand of the skeleton, such as the stem and each node. These strands are then combined into a single field using smoothed minima. Additionally, our tool will employ a technique that controls the smoothing minima to adjust the balance of smoothness and reproduce the distortion effects observed around the dead knots.

We plan to integrate our tool as a plug-in into Maya, which is currently very popular modeling software and is widely used by industry, gaming, and animation communities. Integrating our tools into Maya will allow artists to use our tools more easily without changing their existing workflow. To develop our tools, we will use MEL to create interfaces and handle basic tasks such as importing the skeletal structure of tree logs, providing controls that allow artists to adjust the size, shape, and distribution of knots, and more. We will use the Python API to provide the underlying functionality.

In the first phase, we will conduct a feasibility study focusing on learning to understand the algorithms in the paper for simulating wood and knot textures, and first implement the algorithms in the shadertoy and test the results. After reaching a fair degree of completion, we will start trying to integrate our tools into Maya. This part will include the design of the user interaction interface, beautification, and support for the underlying texture generation functionality. The remaining time will be spent debugging the tool, generating textures for different woods, and trying to import the modeling into game engines (e.g. Unity, UE5) for rendering, simulating the real workflow of artists.

Our tool has the potential to be used in a wide range of industries, including architecture, product design, and game development.

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem or Production/Development Need

Knots typically appear on the wooden surface, including building structural components, wall panels, floors, and tabletops. Wood knots are caused by branches that grow from the trunk, leaving traces in the wood. They create complex twists and turns in the otherwise relatively straight stem texture, producing unique ring patterns. Tree knots are a special characteristic of softwood, and the texture of some hardwoods (such as oak) often has knots as well. Currently, the methods used to simulate ring patterns have limitations. For example, while the level set method is suitable for processing complex geometric shapes, including grafting points and dying knots, its computational efficiency on surface textures is low. Procedural textures, on the other hand, have higher computational efficiency, but the simulation of tree knots has not received significant attention.

1.2. Technology

This tool is based on the 2022 SIGGRAPH paper "Procedural Texturing of Solid Wood with Knots" by Maria Larsson, Takashi Ijiri, Hironori Yoshida, Johannes A. J. Huber, Magnus Fredriksson, Olof Broman, and Takeo Igarashi.

This paper describes a method for procedurally generating textures of solid wood with knots, which can be used in computer graphics and virtual reality applications. The proposed method is based on a procedural approach that generates the appearance of knots based on their physical properties and the surrounding wood material. The authors introduce a new model for simulating the shape and distribution of knots in the wood, which is based on a stochastic algorithm that takes into account the growth patterns of the tree branches. The resulting textures are realistic and can be easily customized by adjusting the parameters of the procedural model. The authors also demonstrate how the generated textures can be used to create realistic 3D models of wooden objects with knots. The proposed method offers a promising approach for generating realistic wood textures in computer graphics applications and could be useful for industries such as architecture, interior design, and product design.

We chose this paper as the basis for our Maya plugin because we saw a high demand for efficient generation of realistic wood textures in industries such as industrial design, animation, and gaming. The method described in this article covers parts that current tools have not addressed, such as simulating tree knots. We believe our tool can further enhance the realism of generated wood textures. Additionally, because it fits well with Maya's modeling workflow, it can further improve the efficiency of artists' work.

1.3. Design Goals

1.3.1 Target Audience.

The target audience for this tool includes professionals in industries such as industrial design, architecture, animation, and gaming, who require efficient generation of realistic wood textures.

1.3.2 User Goals and Objectives

Users can utilize this tool to procedurally generate textures of solid wood with knots for use in computer graphics and virtual reality applications. Specifically, they can adjust the parameters of the procedural model to customize the appearance of knots based on their physical properties and the surrounding wood material. They can then use the generated textures to create realistic 3D models of wooden objects with knots.

1.3.3 Tool Features and Functionality

- A stochastic algorithm for simulating the shape and distribution of knots in the wood, taking into account the growth patterns of the tree branches.
- Procedural generation of textures of solid wood with knots, based on their physical properties and the surrounding wood material.
- Realistic and customizable appearance of knots, based on the parameters of the procedural model.
- Compatibility with Maya's modeling workflow, allowing for easy integration into an artist's workflow.

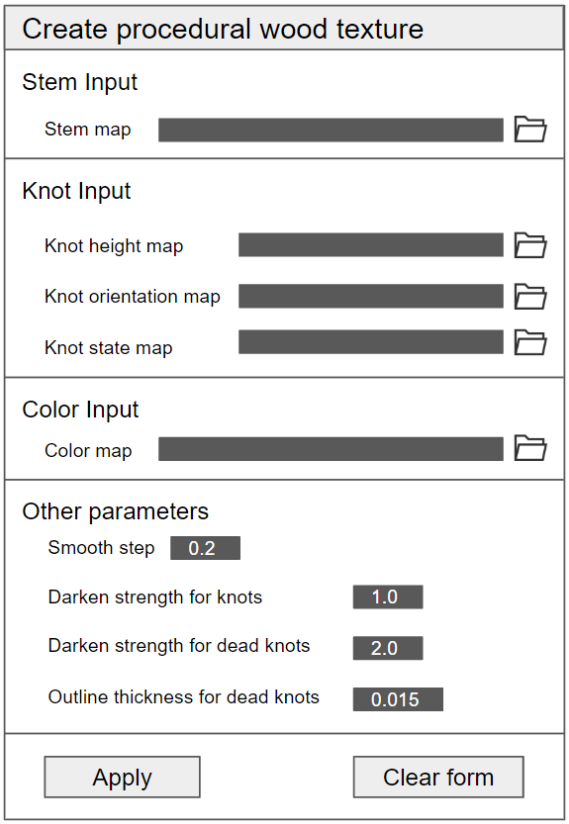
1.3.4 Tool Input and Output

The tool requires a 3D model of the wooden object as input, along with parameters for the procedural model. It produces a procedural texture of the wooden object with knots as output, which can be used to create a realistic 3D model of the object with knots.

1.4. User Interface

1.4.1 GUI Components and Layout

- We offer a user-friendly plugin with a straightforward interface and workflow. Users can easily select their preferred input files for the wood stem, knot, and texture color, and adjust the smooth step for generating the wood texture.



1.4.2 User Tasks

- To use our plugin, users should have a basic understanding of the Maya interface and attribute editing. They can start by creating or importing 3D models and selecting the object(s) they wish to apply the texture to. Our plugin can then be utilized to generate personalized wood textures, allowing the user to configure the stem, knots, and color. Additionally, users can adjust the smoothness between knots and annual rings, the darken strength for knots, the darken strength for dead knots as well as the outline thickness for dead knots.

1.4.3 Work Flow

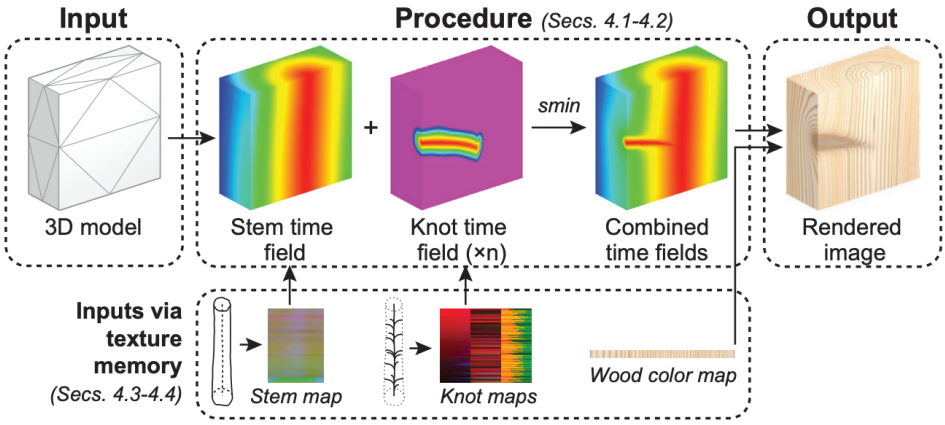
1. Create/import a 3D model.
2. Select the model you want to apply the texture to.
3. Open the procedural wood texture generation tool.
4. Select the input image for the stems and knots.
5. Select a color profile for the wood texture.
6. Fine-tune other parameters such as smoothness, the darkening intensity of the knots, the darkening intensity of the dead knots and the thickness of the dead knot profile.
7. Generate and apply the texture to the model.

2. AUTHORING TOOL DEVELOPMENT

2.1. Technical Approach

2.1.1 Algorithm Details

- Overall
  - The proposed method takes a 3D surface model as well as the internal and external geometry of a tree log as inputs and renders the 3D model with a knotted wood grain texture.
  - The procedure first calculates the **time fields** around each strand, i.e., the stem and each knot.
  - Second, the **time fields** are combined using **smooth minimums** while considering whether each knot is alive or dead at the given point in time.
  - Third, the time values are translated into **wood colors** by sampling a wood color map and additionally darkening the knots.
  - Finally, we present the input tree log data and how they are encoded into images.

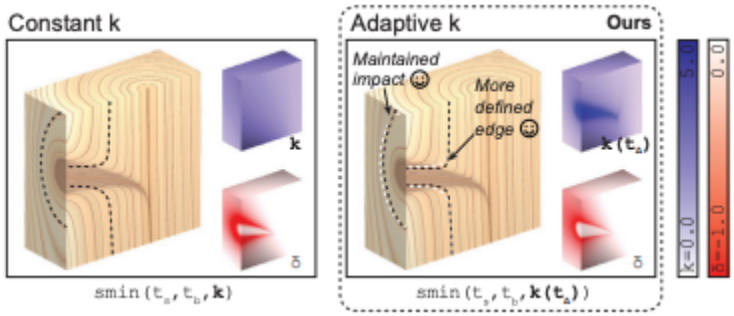


- Representation of annual rings
  - Time field
    - Definition
      - We define the volumetric texture of the wood grain using a 3D scalar field representing the time when the growth occurred, which we refer to as a time field.
    - Calculation

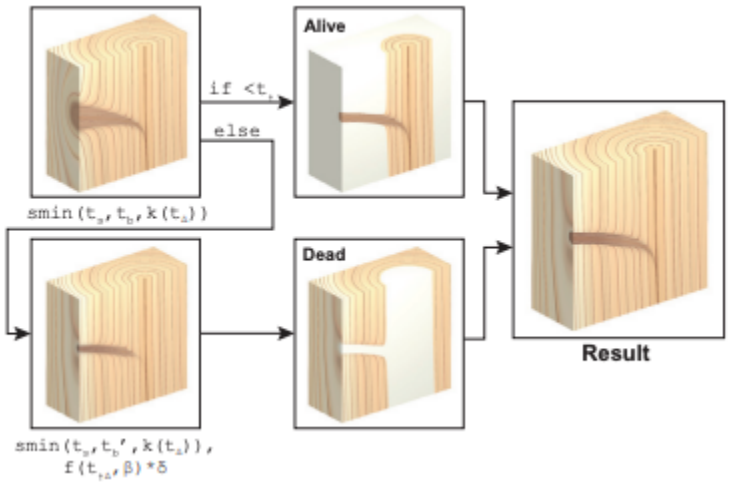
- $t = \frac{dist(S, P)}{v(S, \beta)}$
- time value (t)
- $P \in \mathbb{R}^3$, the world coordinates of a point on a 3D model
- $S \in \mathbb{R}^3$ is the point on the strand nearest to P
- $v(S, \beta)$ is the speed of growth at point S on the stand in the orientation β of P around the strand
- For better performance, we apply a simplification to take a pseudo-nearest point
  - For the stem, we define the pseudo-nearest point as the point on the stem skeleton with equal height (z) as P
  - For knots, the pseudo-nearest point on the branch skeleton is the point at an equal distance (d) from the stem as P.

- <u>Stem-to-knot transition</u>
  - power smooth minimum
    - used to create unions with a continuous seam between implicitly defined surfaces
    - $$smin(t_a, t_b, k) = \left(\frac{t_a^k t_b^k}{t_a^k + t_b^k} \right)^{\frac{1}{k}}$$
      - where *t_a* and *t_b* are scalar values at a point in the field, and k is a parameter that controls the degree of smoothness.
  - fine-tune the shape of the smooth union by adapting the parameter value(k) within a field
    - amount of smoothing (δ)
      - $\delta = smin(t_a, t_b, k) - min(t_a, t_b)$
      - This value (δ) is useful for the analytical purpose of visualizing the amount of smoothing
    - adjust the shape of the smooth minimum by modifying δ
      - $smin'(t_a, t_b, k) = min(t_a, t_b) + \delta'$
- <u>modelling an alivce knot</u>

- When joining the time fields of a knot and the stem by the power smooth minimum, it naturally creates the appearance of an alive knot: the thickness of the knot increases with time and the knot is intergrown with the stem. we set a higher *kb* inside the knot (for a tighter edge) and a lower *ks* outside the knot (for a stronger distortion effect). We create a smooth transition between these two *k*-targets based on the variable $t\Delta$, which is the signed difference between the time values of the stem (*ts*) and knot (*tb*). The result is a more defined knot edge with a maintained distortion effect compared to a constant *k*

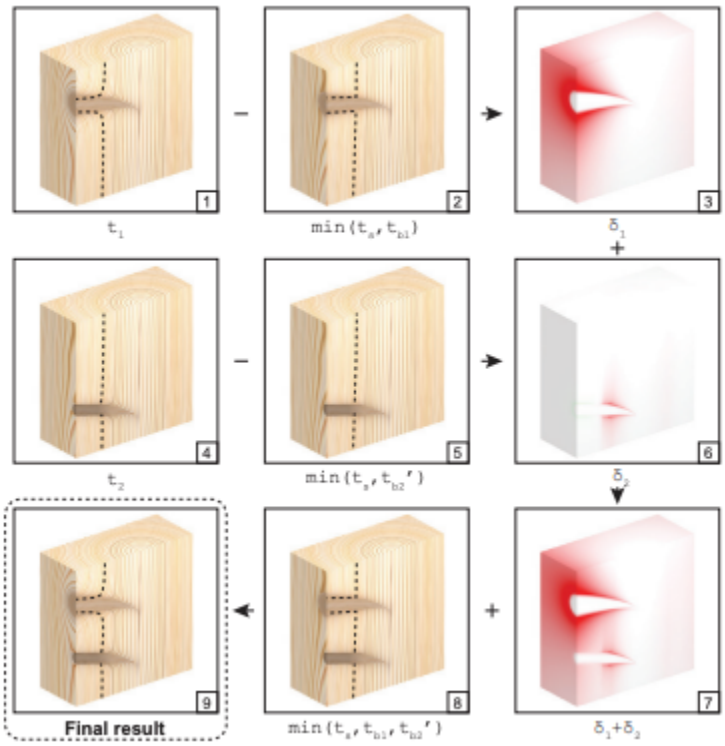


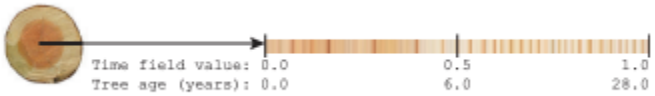
- ○ <u>modelling a dead knot</u>
  - the amount of smoothing (δ) is modified by scaling it by a factor *f* . When scaled by a negative value, the smoothness inverts. The factor *f* is calculated as a function of two variables: time since death ($t\dagger\Delta$) and orientation around the knot axis (β). The time since death ($t\dagger\Delta$) is used as a variable to gradually change the direction of the smoothness with increased distance from the point of death. The orientation (β) is used to create a non- uniform deformation around the knot axis to produce a butterfly pattern.



- The value of interest for the next step in the calculation (joining multiple knots to the stem, Sec. 4.2.3) is the modified amount of smoothing (δ'), the complete calculation of which is summarized as

- $$\delta' = \begin{cases} smin(t_s, t_b, k(t_\Delta)) - min(t_s, t_b), & \text{if alive} \\ f(t_{\dagger_\Delta}, \beta) \cdot (smin(t_s, t'_b, k(t_\Delta)) - min(t_s, t'_b)), & \text{otherwise} \end{cases}$$

- multiple knots
  - calculate the modified smoothing value ($\delta i'$) for each pairwise i-th knot and stem union separately, as shown in Eq. 5 and add their sum to the regular minimum of all knots
    - $$smin'(t_s, t_{b_1}, ..., t_{b_n}) = min(t_s, t_{b_1}, ..., t_{b_n}) + \sum_{i=1}^{n} \delta'_i$$
    - 

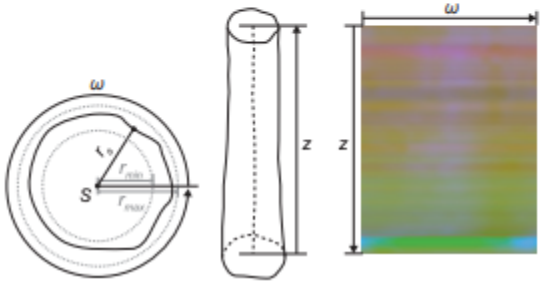
- color rendering
  - To translate the time field into wood colors, we map it to a one- dimensional wood color map
    - 

- input tree log data
  - The input tree log data consist of its internal grafting skeleton, its outer surface, and the point of dying of each knot. These data are encoded in image

files, which are uploaded to the texture memory and sampled during the procedure.

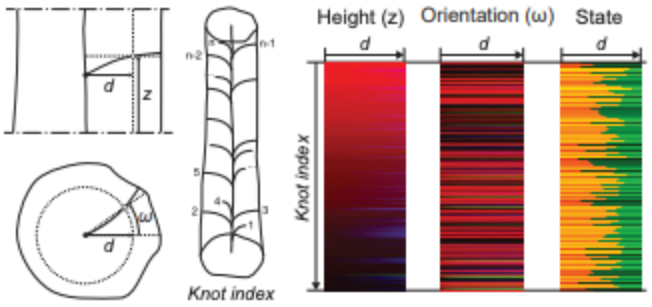
- stem geometry
  - The skeleton strand and outer surface of the stem are encoded in an image, referred to as the stem map
  - The horizontal image axis corresponds to the ori- entation around the tree (ω), and the vertical axis corresponds to the height inside the tree (z).
  - We currently use an RGB texture image with a pixel resolution of 128 × 512 for the stem geometry.

Image		Data	Range
Axis	**X**	Rotation (ω)	0.0 to 2π
	Y	Height (z)	0.0 to h
Color	**R**	Pith point x (S_x)	$-0.5 \cdot r_{s_{min}}$ to $0.5 \cdot r_{s_{min}}$
	G	Pith point y (S_y)	$-0.5 \cdot r_{s_{min}}$ to $0.5 \cdot r_{s_{min}}$
	B	Radius (r_s)	$r_{s_{min}}$ to $r_{s_{max}}$



- knot geometries
  - We encode the geometries of the knots in three image maps: the knot height, knot orientation, and knot state maps
  - For these maps, the horizontal image axis corresponds to the distance from the pith (d), and the vertical image axis to the knot index. For a particular knot index and distance from the pith, the knot skeleton point is given by its height in the tree (z) and orientation around the vertical axis (ω), which are sampled from the knot height and knot orientation maps, respectively.
  - The third map—the knot state map—indicates when the knot is alive or dead.

Image			Data	Range
All	**Axis**	**X**	Pith distance (d)	0.0 to $r_{s_{max}}$
		Y	Knot index	1 to n
Knot height map	**Color**	**R**	Start (z_0)	0.0 to h
		G	Increase (z_+)	0.0 to $r_{s_{min}}$
		B	Decrease (z_-)	0.0 to $r_{s_{min}}$
Knot orientation map	**Color**	**R**	Start (ω_0)	0.0 to 2π
		G	Left (ω_{ccw})	0.0 to 0.5π
		B	Right (ω_{cw})	0.0 to 0.5π
Knot state map	**Color**	**R**	Alive	T/F
		G	Time of death ($t_\dagger$)	0.0 to 1.0

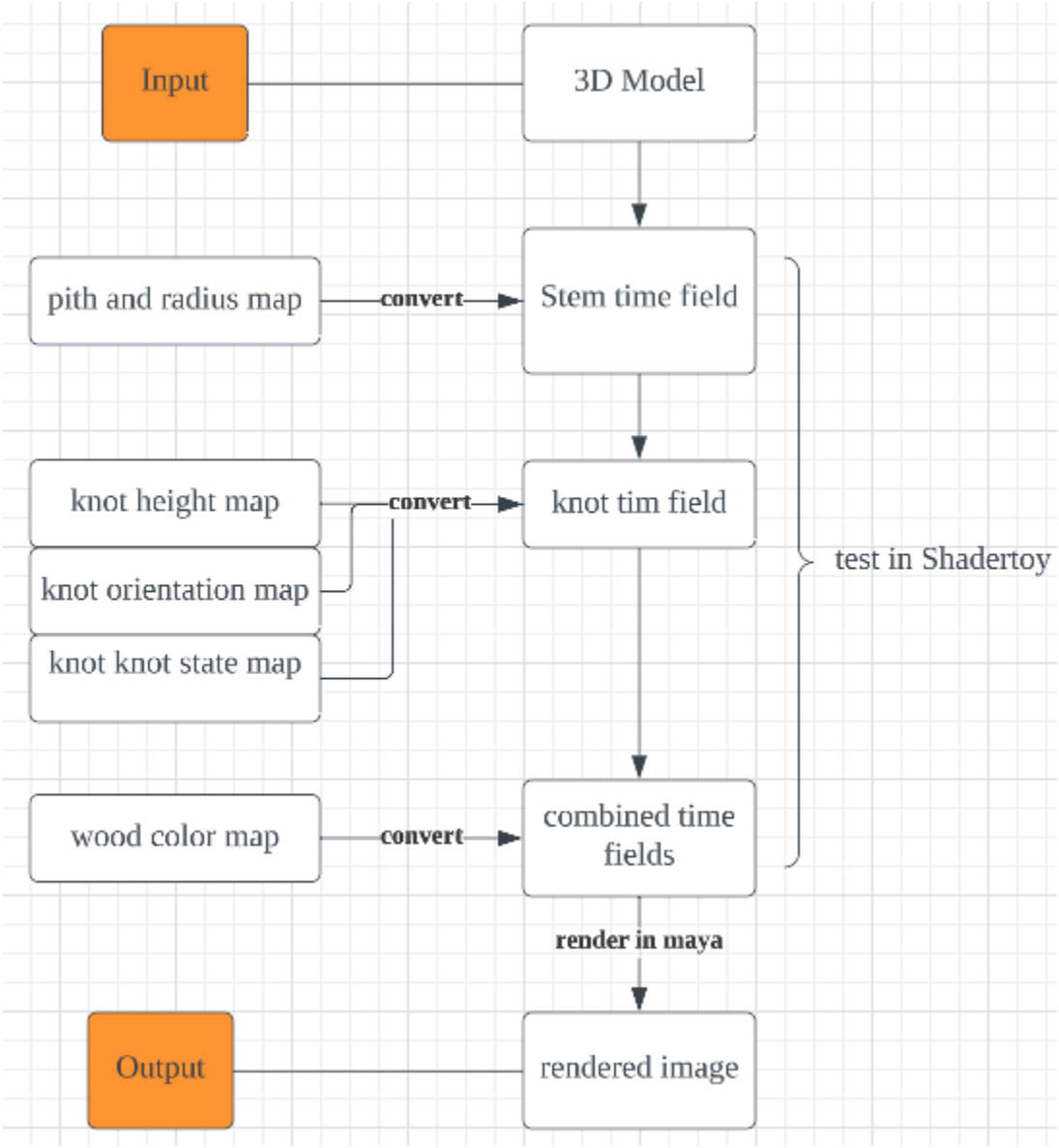
- 

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. The process will be written in OpenGL Shading Language(GLSL) and the setting up part will be written in Python. Specifically, users will be able to apply the knot texture with the MEL command “apply texture of solid wood with knot”, which will apply the texture on to the selected mesh
- the tool will be using these Maya objects and classes:
  - MfnMesh – is Maya’s polygonal mesh object
  - MViewportRenderer – allows control over drawing and assigning textures to objects in a scene
  - MObjectArray – General data structure for storing various objects or textures

2.1.3 Software Design and Development

- The process will be written in OpenGL Shading Language(GLSL). The core data structures for calculating the tree knots are given below:
  - knot_height_map & knot_orientation_map:
    - For a particular knot index and distance from the pith, the knot skeleton point is given by its height in the tree (z) and orientation around the vertical axis (ω), which are sampled from the knot height and knot orientation maps, respectively.
  - knot_state_map:
    - indicates when the knot is alive or dead.
  - pith_and_radius_map:
    - The skeleton strand and outer surface of the stem are encoded in an image, referred to as the stem map
  - wood_bar_color & wood_bar_normal & wood_bar_specular
    - To translate the time field into wood colors, we map it to a one-dimensional wood color map
  - main.vert & main.frag
    - we won't do anything special in the vertex shader
    - the procedure is located in the fragment shader



2.2. Target Platforms

2.2.1 Hardware

- (32-bit) Intel® Pentium® IV or higher, AMD Athlon® XP processor (must be SSE capable)
- 1 GB RAM
- Hardware-accelerated OpenGL® graphics card

2.2.2 Software

- (32-bit) Windows XP Professional (SP2)

- Maya 8.5
- OpenGL 1.2+
- Python 3.8.7
- pip 20.2.3
- glfw 2.1.0
- numpy 1.20.3
- Pillow 8.2.0
- PyOpenGL 3.1.5
- pyrr 0.10.3
- openmesh 1.2.1

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

- Due March 27th: Knot texturing will contain UI, load input image and implement both alive knot and dead knot in the texture

2.3.2 Beta Version Features

- Due April 12th: Wood Knot Editor will contain UI, ask the user for tree information(degree of smoothness step k, darken strength for knots, darken strength for dead knots, adn outline thickness for dead knots) and implement both alive knot and dead knot in the texture

2.3.3 Description of any demos or tutorials

- Alpha demo: Wood Knot Editor will display UI and be able to load the input maps needed and apply alive knot texturing on the mesh selected. The alpha demo will show the tool workflow and hint at the look and performance of the final plugin.
- Beta demo: Wood Knot Editor will display UI and be able to apply both alive and dead knot texturing on the mesh selected. The user can modifies parameters on the textures.
- Final demo: Wood Knot Editor will fix bugs and address comments from the class.
- Tutorial: The final project will include code, sample source data, and how-to-document. Users will be able to run the final demo out of the instruction.

3. WORK PLAN

3.1. Tasks and Subtasks

Our tool's development has been segmented into distinct tasks.

3.1.1 Task 1 - User Interface (Wenqing) 3.13 - 3.17

This task involves creating the Maya User Interface. This will entail designing the necessary GUI dialogs for the tool using MEL. This involves the following subtasks:

3.1.1.1 Subtask 1 - Layout Design (Wenqing) 3.13 - 3.15

Designing the layout of the GUI, deciding the placement of different UI elements such as buttons, sliders, and other controls. We will design the main window, file loader dialog, and parameter input box, while also assessing various options to ensure maximum usability.

3.1.1.2 Subtask 2 - Prototyping (Wenqing) 3.16 - 3.17

Creating prototypes of the GUI to test its functionality and usability.

3.1.2 Task 2 - Data Acquisition (Xu) 3.13 - 3.17

This task involves acquiring data of the wood's texture (including stem textures and knots textures) and color that will be used to create procedural textures. For this step, we'll first use the provided textures in the open-source project and then create variations by ourselves.

3.1.3 Task 3 - Procedural Texture Generation (Wenqing, Xu) 3.20 - 3.31

This task involves generating procedural textures based on the acquired data. This involves the following subtasks:

3.1.3.1 Subtask 1 - Generate textures with hard-coded parameters (Xu) 3.20 - 3.24

In this step, we will implemented the main algorithms presented in our reference paper, including the min smooth step function and the creating of basic annual ring texture and the combination with knots. Since we won't use any file inputs here, we plan to complete this task in ShaderToy.

3.1.3.2 Subtask 2 - Use input data to generate textures (Wenqing) 3.20 - 3.24

After validating the main algorithms in ShaderToy, we'll then use texture images as input to set the wood stem and knots. We'll use Python to implement the entire pipeline.

3.1.3.3 Subtask 3 - Add user-defined parameters 3.27 - 3.31

(Wenqing, Xu)

We'll expose more parameters such as the smoothness between knots and annual rings, the darken strength for knots, and etc. that allow our users to fine-tune the final results.

3.1.4 Task 4 - Integrate basic fratures into Maya (Wenqing, Xu) 4.3 - 4.7

Integrating the algorithms into the GUI so that users can create procedural textures using the GUI.

3.1.5 Task 5 - Testing and Exporting (Wenqing, Xu) 4.10 - 4.14

The final step involves exporting the results of the tool. This involves the following subtasks:

3.1.5.1 Subtask 1 - Generate various textures in Maya (Xu) 4.10 - 4.12

For this subtask, we will test our plug-in by using different input log files and different parameters to generate all-kinds of wood textures.

3.1.5.2 Subtask 2 - Export texture in game engine (Wenqing) 4.13 - 4.14

For this subtask, we will export the generated procedural textures in a format that can be used by other software such as Unity3D and Unreal Engine 5. We will also try to export the 3D models with the generated textures applied to them.

3.2. Milestones

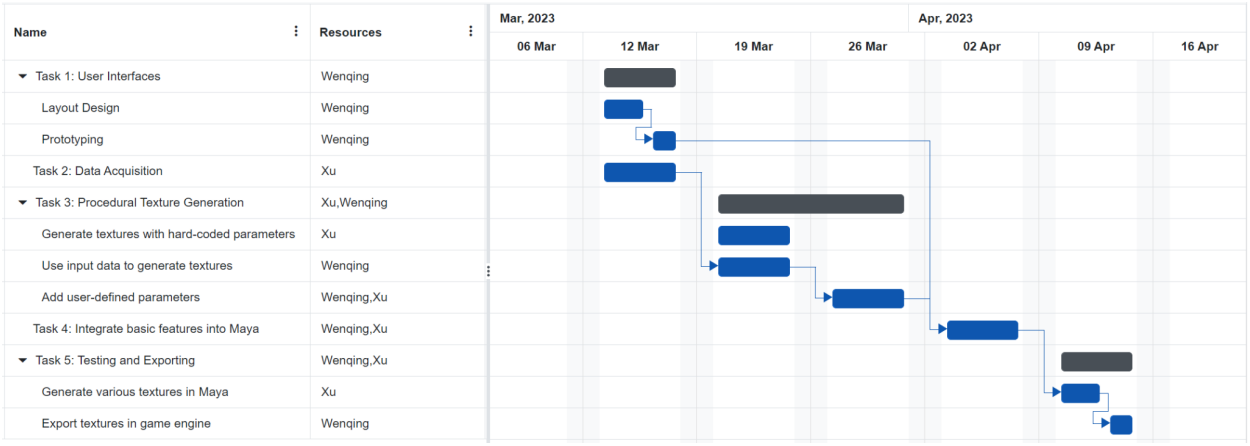
3.2.1 Alpha Version

Alpha version will complete its corresponding tasks 1, 2, 3.1.3.1, 4

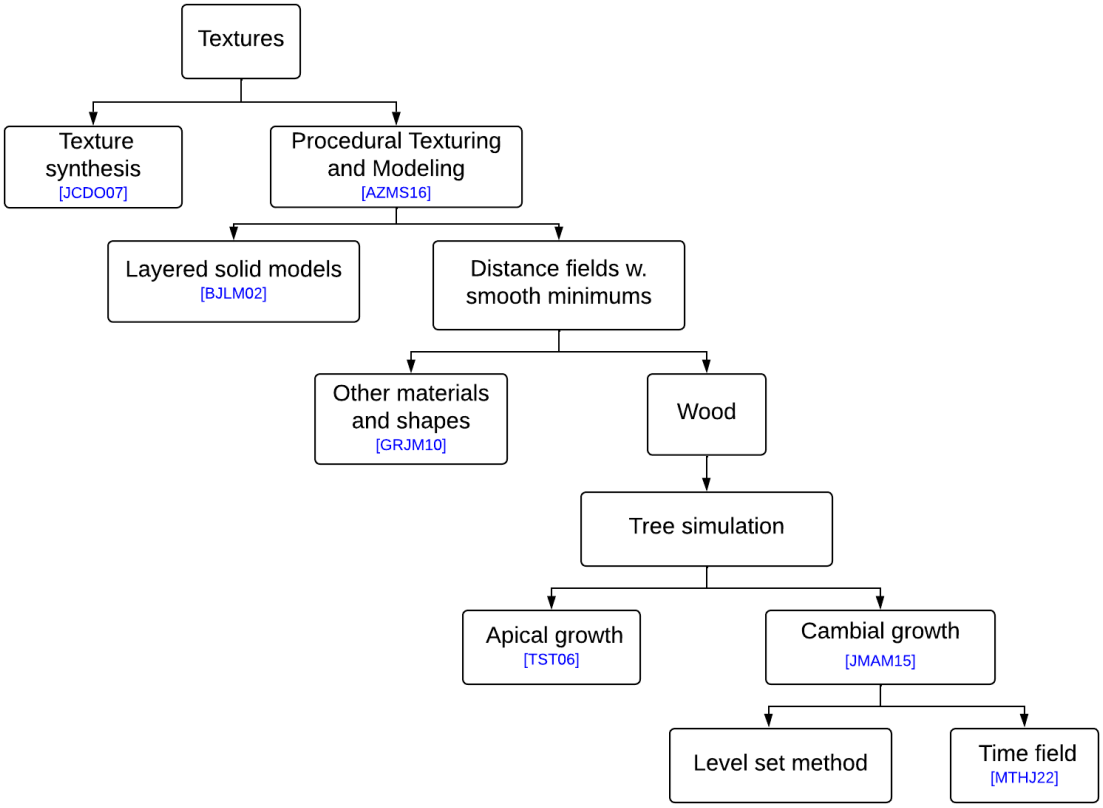
3.2.2 Beta Version

Beta version will complete its corresponding tasks 1, 2, 3, 4, 5

3.3. Schedule


Name
Resources
Mar, 2023
Apr, 2023
06 Mar
12 Mar
19 Mar
26 Mar
02 Apr
09 Apr
16 Apr
Task 1: User Interfaces
Wenqing
Layout Design
Wenqing
Prototyping
Wenqing
Task 2: Data Acquisition
Xu
Task 3: Procedural Texture Generation
Xu,Wenqing
Generate textures with hard-coded parameters
Xu
Use input data to generate textures
Wenqing
Add user-defined parameters
Wenqing,Xu
Task 4: Integrate basic features into Maya
Wenqing,Xu
Task 5: Testing and Exporting
Wenqing,Xu
Generate various textures in Maya
Xu
Export textures in game engine
Wenqing

4. RELATED RESEARCH



[JCDO07] The paper "Solid texture synthesis from 2D exemplars" presents a method for synthesizing 3D textures that can be applied to a variety of surfaces, including both geometric and organic shapes. The proposed method uses a 2D texture exemplar as input and generates a 3D texture that closely matches the visual appearance and spatial characteristics of the exemplar.

[AZMS16] This paper presents a method for simulating the appearance of wood by modeling its complex structure and texture. The method includes a novel algorithm for simulating the growth rings and fiber orientations found in real wood, as well as a technique for modeling the effects of light scattering and absorption within the wood material to simulate the wood's texture. The proposed

method is evaluated using both quantitative and qualitative metrics and has been shown to produce realistic wood textures that closely resemble the appearance of real wood.

[BJLM02] This paper presents a new procedural modeling system that allows for the creation of complex 3D models through the specification of procedural rules and constraints. The proposed system allows for the creation of models with a high degree of detail and can be used to model a wide range of objects and materials. The system is based on a set of procedural modeling operators that allow for the specification of various geometric and physical properties of a model, including shape, texture, and material properties. The proposed approach allows for the generation of models that are highly customizable and can be adapted to a range of applications, including architectural design, product design, and digital fabrication. The paper includes an evaluation of the proposed system, which demonstrates its ability to create complex models with a high level of detail and realism.

[GRJM10] In this paper, the authors develop a new procedural texturing approach that uses geodesic distance fields to generate textures for complex meshes. The proposed approach allows for the creation of realistic and highly detailed textures that can be adapted to a wide range of 3D models. The paper introduces the concept of geotextures, which are textures generated by computing geodesic distances to various features on a mesh surface. The

proposed approach uses a set of operators to generate geodesic distance fields, which are then combined to produce textures with different visual characteristics, such as color, roughness, and displacement.

[TST06] This paper proposes a new system for generating tree models using free-form strokes. The proposed Sketch L-system allows for the creation of tree models with a high degree of control and flexibility, using user-defined strokes to guide the growth and shape of the tree. The paper combines the traditional L-system approach with free-form strokes to create a more intuitive and user-friendly modeling system named Sketch L-system. The proposed system allows users to draw strokes on a 2D canvas, which are then used to guide the growth and shape of the tree in 3D space.

[JMAM15] The main contribution of the paper is the development of a new system for simulating the growth of trees and generating realistic tree models called Modification. It allows users to interactively control the growth of a tree and produce models with a high degree of realism and detail. The paper introduces the concept of cambial growth modeling, which is used to simulate the growth of trees by modeling the behavior of cambium, a layer of cells in the trunk of the tree that is responsible for the growth of new tissue. The proposed system allows users to control the growth of the tree by drawing curves on the surface of the trunk, which is then used to guide the growth of the cambium.