Authoring Tool Design Document Outline: Chimera

Hannah Bollar and Wenli Zhao

CIS 660: Advanced Topics in Computer Graphics
Instructor: Dr. Stephen Lane
February 27, 2019

[This page is the back of the title page. It is intentionally left blank.]

PROJECT SUMMARY

The concept of zoomorphic design is the application of animalistic features on non-animal objects. These shapes are commonly used in furniture, interior and exterior structures, general household objects, and character designs. They have a unique aesthetic and bring a sense of levity and detail to the originally blunt creation, due to humanity's affinity towards and common comfort found in friendly cuddly figures. These designs can be hard to image conceptually and even to create physically using traditional modeling tools. This difficulty is because of the inherent complications that arise in balancing the underlying structural design with the added organic figure while modeling the final product mesh by hand.

The paper on which this tool is based is the seminal paper for zoomorphic design in graphics. Though previous techniques and research have been used for three-dimensional shape modification, architectural creation and synthesis, and general mesh composition, this paper is the first in computer graphics that actually develops a complete computational approach that combines mathematical and structural reasoning for the full zoomorphic process. Thus, this will be the first plugin application available to the public that implements this novel operation.

In terms of design goals, this tool is aimed to be as easy to interface with as possible, such that individuals can enjoy its novel outputs regardless of their technical backgrounds.

Since this product has a niche type of output, it will have a bit more niche of a user base. As a focus, it will be used more commonly by general artists and designers such as for architectural creations, playground environments, children's toys, and any generic interesting shape construction. This program is beneficial for these cases as not only will it be generally speeding up the construction of the final product, but also it allows for multiple iterations of possible proposed beginning combinations of the two inputted shapes.

The tool will function as a Maya plugin that the user will interact by selecting the two desired meshes and labeling them for the plugin as either the base or animal mesh. Once these inputs are added, the user designates what sections of the base, if any, need to be structurally maintained by a volumetric selection. Lastly, the user has a slider bar for manipulating the interpolation between how much the output forms to the base shape and conversely how much it allows the animal shape to take over the output creation.

The implementation will specifically be with a C++ base and MEL interaction. We will first be testing our functionality separate from the plugin in an OpenGL setup, and once we've confirmed it works successfully, we will import it into Maya as an official plugin.

For our development schedule, we plan to have our base OpenGL implementation functioning by Alpha; adding in volumetric restriction, more user inputs, and porting over to Maya by Beta; and to only need to do final testing and debugging between Beta and Final submission. During this last section, we'll also be taking the time to come up with as many novel outputs that would be fun and interesting representations of the functionality of our plugin.

We hope you enjoy using our tool!

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem or Production/Development Need

This authoring tool aims to automate parts of the 3D modeling process specifically for the use case of creating 3D zoomorphic designs. It addresses the need for easier 3D modeling techniques. It also addresses the development need of creating Zoomorphic designs. From product design to toys and character design, Zoomorphism is everywhere in man-made design, however it is not a trivial task to create a convincing or functional Zoomorphic object. The tool allows designers to easily create different zoomorphic designs without having to manually model a final design. The designer only needs input models and a general vision of their output. Since input models are common objects and animals, these should be very accessible to find as designers. This allows designers to experiment with various designs without having to go through a lot tedious modeling.

1.2. Technology

Throughout history man-made objects have been inspired by animal forms. The authors of the paper “Zoomorphic Design” note that Zoomorphism is prevalent in architecture, furniture and product design as well as children’s toys and decided to tackle the unique design challenges of Zoomorphic design through a computational approach.

The paper describes a method for combining the meshes of man-made objects and animal objects to create novel 3D meshes. Their technique ensures the final zoomorphic shapes do not violate design restrictions and provides a versatile method for generating a wide variety of shapes. The paper uses techniques from 3D shape modeling, 3D optimization and analysis as well as Computational design.

They use shape graphs to segment models in databases of shapes and suggest similarity scores based on a graph kernel for the objects. This allows suggestion of similar shape pairings for zoomorphic design. They also developed a novel technique called Volumetric Design Restriction. This is a method for labeling the base shape with restricted and free zones to maintain final volumes in the output.

Both the base shape and animal shape are deformed but are constrained to preserve segment adjacencies in the shape graph. They are also optimized for configuration energy, a measure of how desirable the output shape is. The energy is minimized in a coarse-to-fine manner. They are coarsely aligned and then refined with continuous optimization. Finally, unwanted geometry is removed and the resulting shape is a union operation of the two shapes.

The paper also describes goals for potential user control.

We chose this paper because we realized it would make an interesting authoring tool that has a clear user in mind. The process it simplifies is clearly defined and there is lots of room for user control as a tool. We also thought the concept was fun and imaginative and inspires creativity for the end user.

1.3. Design Goals

The authoring tool is aimed artists and designers with little to no technical background but can scale, translate and rotate objects in Maya.

1.3.1 Target Audience.

The target audience of the tool will be artists and designers. This includes concept artists, character designers, product designers, furniture designers, toy designers and set artists. They do not have to understand the intricacies of 3D modeling.

1.3.2 User goals and objectives

We envision users to use the tool for both concept development and asset design. The tool can aid artists and designers in the development phase since it is an efficient way to create novel meshes and designs at a low cost to the artist. Since the process is semi-automated they can easily create 3D zoomorphic designs simply for brainstorming concepts and designs. This allows them to explore more ideas more quickly. The tool will also allow refinement for a final design such as a toy, piece of furniture or 3D asset for a game or animation. The tool is designed for Zoomorphic design, but could potentially be extended to mesh creation from any two inputs.

1.3.3 Tool features and functionality

List and describe the feature set of the tool.

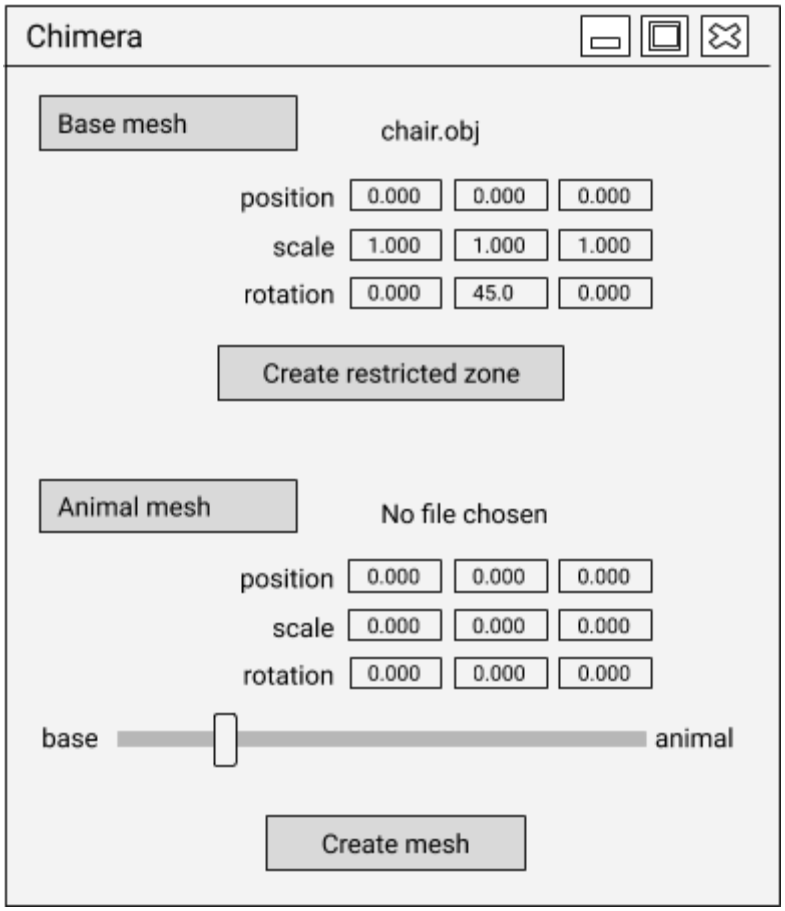
- User has ability to select meshes from a database of 3D models
- User can choose a base mesh and an animal mesh from the database
- User can choose a weight via a slider that influences the final mesh more towards the base shape or the animal shape
- User can choose zones of Volumetric Restriction through mouse input

1.3.4 Tool input and output

The tool will take in two meshes, an initial configuration, a weighting scheme and zones of volumetric restriction as an input. Based on these inputs, the plugin will optimize the two meshes for configuration energy which ensures a desirable shape and apply input restraints and other refinements to create a novel mesh.

1.4. User Interface

1.4.1 GUI Components and Layout



[Proposed GUI Design]

- The user will choose a base mesh file and upload it and can edit the configuration of the mesh via the GUI or Maya's UI
- They can then choose to create restricted zones of the base mesh. Clicking the button will allow them to select, via Maya, parts of the mesh they want to restrict to the base mesh form. (This will be a stretch goal)
- The user must also choose an animal mesh
- The user can then use a slider to choose how much influence the base mesh has or the animal mesh has on the final mesh.
- The meshes will be visible in Maya and the user can manipulate both or replace either by changing the browsed file location if they choose to load a new base mesh or animal mesh.

1.4.2 User Tasks

- The user must choose the base mesh and the animal mesh
- The user must position the two meshes in a desired configuration so the two meshes are roughly aligned.
- The user must understand the concept of Volumetric Design Restriction to create a restricted zone. They must understand that by creating a restricted zone, they will see a volumetric rendering of what areas in the base mesh must be maintained in the final novel mesh.
- The user should creatively understand that weighing the tool towards the base mesh preserves more of the base mesh shape whereas the animal shape will preserve the animal shape

1.4.3 Workflow

- The user will open the tool in an empty Maya scene
- They will first select a base mesh they want to use from a database of prelabeled meshes. e.g. a 3D mesh of a chair.
- They can then choose a restricted zone if they want to. For example, they will click on the seat of the chair and mouse over it to create a volumetric restricted area. They will see a preview rendering of the restricted zone. This ensures that this area of the base mesh will not be deformed or removed in the final mesh.
- The user will then select an animal mesh.
- The user can move or transform the animal mesh via the GUI or Maya editor to align with the base mesh.
- The user uses a slider to determine how much they want to preserve the base shape vs. the animal mesh shape
- Finally, the user can click create mesh which will generate the novel 3D mesh
- Rather than modeling everything by hand, the tool automatically generates a 3D mesh that is a merged form of the base mesh and an animal mesh.
- The inputs are the desired meshes to be combined, potentially a zone on the base mesh for restricted volumetric design, a scalar value for weighting the meshes, and a configuration of the two meshes.

2. AUTHORING TOOL DEVELOPMENT

2.1. Technical Approach

2.1.1 Algorithm Details

Describe the main features and details of the algorithms you plan to implement.

- Preprocessing step
  - Input data
    - two input meshes: base shape, M_b, and animal shape, M_a
    - input shapes are upright
    - segment all shapes into semantically meaningful parts (e.g. head, neck, torso, upper arm, etc.)
    - This feature will require a user tool that allows us to paint parts of each mesh for preprocessing
  - Annotation
    - each base shape is annotated with volumetric design restriction labels (see more info below)
    - each animal shape is annotated with visual salience labels
      - used to identify which regions of animal shape should be visible in the output Zoomorphic shape, (e.g. ensure we preserve the shape of a horse's head on the output shape)
- Volumetric Design Restriction
  - label the mesh of the base shape with VDR, ensures that the animal shape will not violate the design restrictions to preserve certain features of the base shape (e.g. hollow interior of a mug, roundness of wheels on a car)
  - All volume around the shape will be categorized as either *free* or *restricted.* An animal mesh can lie in the free zone, but not the restricted zone.
    - *restricted zone:* a zone a volume where the animal mesh would violate the design restriction of the base mesh.
    - *free zone:* all other zones.
    - To determine what is free and restricted, we label the faces on the base shape into 4 different categories which include:
      - Infinite Free: all space around the section of the surface is free. (e.g. back of a chair)
      - Infinite Restricted: all space around the section of the surface is restricted
      - Finite Free: Space within finite distance is free, beyond a distance it is restricted (e.g. chair legs)
      - Finite Restricted: space within a finite distance is restricted, this distance it is free

        - each face is assigned a label, α, which denotes whether it is free or restricted and β, which is a real number that denotes a threshold from the face.
        - Note that space inside the base shape is free because it is already occupied by one shape, so it is free to be occupied by another shape.
    - For any point, p, in the space, we can determine whether it is free or restricted
        - determine the closest face on the base mesh to the point
        - compute signed distance, γ, from the base mesh to the point
        - $$r(\mathbf{p}) = \begin{cases} 1 & \text{if } \gamma \leq 0, \\ \hat{\alpha} & \text{if } \gamma > 0 \text{ and } \gamma < \hat{\beta}, \\ -\hat{\alpha} & \text{if } \gamma > 0 \text{ and } \gamma \geq \hat{\beta}, \end{cases}$$
        - r(p) = -1 corresponds to the restricted zone and r(p) = 1 is free
        - the cases above correspond to the following labels:
	$\beta < +\infty$	$\beta = +\infty$
$\alpha = -1$	*Finite Restricted*	*Infinite Restricted*
$\alpha = 1$	*Finite Free*	*Infinite Free*

- Deformation Models & Configurations
    - Allow deformation of both Base mesh and Animal mesh during optimization process. Optimization is a minimization of configuration energy (see next section)
    - We will employ one of two mesh deformation models depending on the mesh that is chosen:
    - Linear Blend Skinning: mesh input must come with skeleton & weights, suitable for certain meshes.
    - Free Form Deformation
    - Base Shape Deformation
        - can be transformed by translation and scale
        - translation is constrained to preserve segment adjacency
        - groups of segments can be constrained to have same transformation (e.g. preserve symmetry and functionality)
        - scales in different directions can be constrained to be equal, for example, to preserve the roundness of a wheel
    - Configuration
        - define a configuration $\phi = (\phi_a, \phi_b)$ where ϕ_a and ϕ_b are vector of configuration parameters of the animal shape (a) and base shape (b)
- Configuration Energy
    - configuration energy allows us to measure of the desirability of the zoomorphic shape resulting from a given configuration
    - A desirable zoomorphic shape should have a **low** energy

- We want to minimize the following configuration energy:

$$E(\phi, \mathbf{w}) = w_{\text{df}}^{\text{a}} E_{\text{df}}^{\text{a}}(\phi_a) + w_{\text{df}}^{\text{b}} E_{\text{df}}^{\text{b}}(\phi_b) + w_{\text{r}} E_{\text{r}}(\phi) + w_{\text{vs}} E_{\text{vs}}(\phi) + w_{\text{g}} E_{\text{g}}(\phi),$$

  - $\mathbf{w} = [w_{\text{df}}^{\text{a}}, w_{\text{df}}^{\text{b}}, w_{\text{r}}, w_{\text{vs}}, w_{\text{g}}]^T$ is a vector of weights, for the initial implementation, they will be set to 1.0, which should be sufficient for fully automatic operations
  - The five individual terms in the equation include
    - Animal shape deformation: we penalize the animal shape deformation. If any skeleton bone is stretched more than a threshold or the angle between a pair of bones differs more than the rest angle by a threshold, the deformation is invalid

$$E_{\text{df}}^{\text{a}}(\phi_a) = \frac{D(\phi_a)}{D_m} + C(\phi_a),$$

    - Base shape deformation: we penalize non-uniform scaling of the base shape, this term is a sum of squared differences of all pairs of segment scales of the base shape segments

$$E_{\text{df}}^{\text{b}}(\phi_b) = \frac{1}{K_{\text{df}}^{\text{b}}} \sum_{i<j} \sum_{u,v} (s_{i,u} - s_{j,v})^2,$$

    - Registration: a term to encourage animal shape and base shape to align with each other. Computed over a set of uniformly sampled vertices from the base mesh

$$\begin{aligned} E_{\text{r}}(\phi) &= 2 - \frac{1}{|\mathcal{V}_{\text{r}}|} \sum_{\mathbf{v}} \exp\left(-\frac{d(\mathbf{v})^2}{{\sigma_{\mathbf{v}}}^2}\right) \\ &- \frac{1}{|\mathcal{V}_{\text{r}}|} \sum_{\mathbf{v}} \exp\left(-\frac{\arccos(\mathbf{n}_{\mathbf{v}} \cdot (\nabla d)_{\mathbf{v}})^2}{{\sigma_{\mathbf{n}}}^2}\right), \end{aligned}$$

    - VIsual salience: this term encourages appearance of visually salient regions from the animal shape in the final zoomorphic shape

$$E_{\text{vs}}(\phi) = -\frac{1}{\mathcal{A}(\mathcal{V}_{\text{vs}})\,|\Omega|} \sum_{\omega \in \Omega} \mathcal{A}(\mathcal{V}_{\text{vs}}^{\omega}),$$

    - Gash: Restricted zones may remove parts of the animal shape which will create gashes on the boundary of free and restricted zones. This term penalizes visual gashes.
      - Gashes inside base shape can be ignored since they are concealed
      - Identify a gash surface on the animal mesh
      - Search for pairs of adjacent vertices (m,n) such that m is free and n is restricted. m will be the gash vertex
      - Compute the gash term for the set of all gash vertices

$$E_{\mathrm{g}}(\phi) = \frac{1}{K_{\mathrm{g}}} \sum_{\substack{\mathbf{v}' \in \mathcal{N}(\mathbf{v}) \\ \mathbf{v}, \mathbf{v}' \in \mathcal{V}_{\mathrm{g}}}} ||\mathbf{v} - \mathbf{v}'||,$$

- Zoomorphic Shape Creation
  - Given the animal mesh and the base mesh, we search for the most desirable way to arrange them before merging them. We want to minimize the configuration energy as described in the section above. The shape creation can be summarized by three stages
    1) Coarse stage: find generally good correspondence between two shapes
    2) Refinement: expand degrees of freedom to resolve situations that need more subtle adjustments
    3) Removal and merge: Remove unwanted geometry. Resulting shapes are then merged by a union operation in the volumetric space to create the final zoomorphic design
  - Correspondence Search
    - In the coarse stage, we want to coarsely explore the energy landscape, explicitly associating each correspondence with a configuration
    - Correspondence is a set of pairs of corresponding edges in M_a and M_b, c = $\{(a_i,b_i), i = 1,...,N\}$
    - The goal is to identify a rough pose of M_a that can be refined later
    - We use a combinatorial search tree where c_{child} is equal to its parent node's correspondence plus a new pair of corresponding segments
      - start at root
      - compute configuration energy of each visited node
      - if there exists a node with infinite correspondence (when animation mesh deformation is infeasible) prune this subtree
      - search only considers correspondences that satisfy bilateral symmetry
      - final output is a set of correspondences and their associated configurations
      - select the configuration & associated correspondence with the lowest energy
    - We define ϕ(c) as the configuration associated with correspondence c, i,e., ϕ(c) should deform M_a such that a_i is aligned with b_i
    - To find ϕ(c) first apply local transforms such that each edge in M_a corresponds to an edge in M_b
    - Note that c is a partial correspondence, we might still need to find the deformation of M_a segments that were not in the set c.

                - we find this deformation by minimizing ARAP energy over the entire animal shape with locally transformed segments constrained to be fixed.
            - Output of this stage is set of correspondences and their associated configurations. We select the configuration with the lowest energy and associated correspondence for refinement
        - Configuration Refinement
            - Optimize the full set of control parameters ϕ_a and ϕ_b
            - We minimize configuration energy using Covariance Matrix Adaptation Evolution Strategy and terminate once configuration energy has not decreased by more than 10% in the last 25 iterations
        - Removals and Merging
            - Convert Shapes to their volumetric forms
            - Remove elements from animal mesh if:
                - any of its geometry lies outside of the restricted zones defined by VDR
            - Remove elements from the base mesh if all three qualifications are met:
                - Removing b_i does not reveal a gash on the animal mesh
                - There exists an a_i that replaces the b_i to be removed. (check if a_i is part of a correspondence pair)
                - a_i overlaps with each of b_i's adjacent segments (e.g. horse's leg replacing chair leg should also overlap with chair's base). This overlap is detected using volumetric restriction of the segments
            - Take union of the two volumetric representations as zoomorphic output
            - Apply several iterations of implicit smoothing to the volumetric representation of zoomorphic output where the base and animal shapes intersect
            - volumetric zoomorphic output is converted to a manifold mesh using marching cubes.

Simplifications include:

- not allowing the user to pre-select volumetric restrictions (we have a base algorithm to implement this for now, as a stretch we might implement this as an add-on later).
- not implementing for a database of candidate shape pair suggestions.
- not allowing for segment removal during the optimization process.
- not allowing for user-guided refinement to stop the optimization process early.

2.1.2 Maya Interface and Integration

Describe how you plan to implement the algorithms in the Maya or Houdini runtime environment.

- It will be a Maya Plugin and fileloader using MEL with a C++ backend for the actual algorithm implementation.

What features will be implemented in a scripting language such as MEL, Python, etc.?

- MEL will include all window loading menu interfaces for the user to interact with.
- No aspect of this plugin will be implemented in Python.
- The user does not interact with the actual C++ basis of the plugin.

What features will be implemented in the C++ plug-in?

- The testing phase of this plugin will be implemented in C++.
- The backend feature aspect of the plugin that actually solves the optimizations and shape deformations will also be implemented in C++.

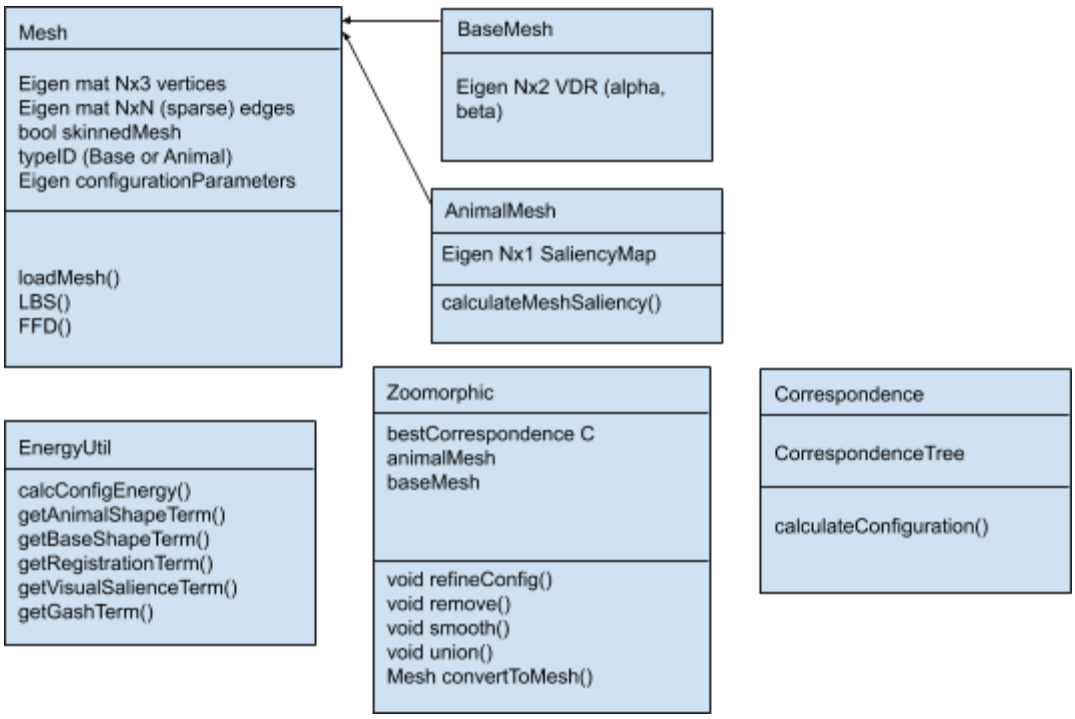
Provide descriptions of Maya objects, data structures and class hierarchies you plan to use, as appropriate.

- MObject - for holding the actual mesh information
- MVector - for transform manipulations by the user for scaling and moving as they “prefit” how they want the objects to try to fit together.

2.1.3 Software Design and Development

Spec out the objects, custom nodes, data structures and class hierarchies you plan to develop as part of the authoring tool implementation.

- We will implement the following c++ classes:
- Mesh: This class will represent the 3D meshes for both Animal meshes and Base meshes. It will be the base class for both types of meshes and contain information about the geometry and functions for mesh deformation.
- Animal: Inherits from Mesh and contains information about saliency and functions for calculating mesh saliency
- Base: Inherits from Mesh and contains class variables to store information about visual design restrictions
- EnergyUtil: Static class to calculate configuration energies
- Zoomorphic: Class that does the main creation of the final mesh. Includes calling optimization iterations and refining volumetric representations of Animal and Base meshes and converting the volumetric representation to a final triangular mesh.
- Correspondence: Stores information of both meshes as a correspondence tree and calculates the configuration energy of a given correspondence



List any third party software you plan to use as part of your development effort.

- Eigen
- Visual Salience Map Creator (maybe, if time we'll write this ourself)
- Linear Blend Skinning
- Free Form Deformation

2.2. Target Platforms

2.2.1 Hardware

Minimum hardware configuration:

- Windows 10
- Intel(R) Core(TM) i7-8550U @ 1.8GHz 1.99 GHz
- 16 GB RAM
- 64 bit OS

2.2.2 Software

Minimum For testing functionality before plugin implementation:

- Windows 10
- Qt 5.9.0
- OpenGL 3.2
- Eigen
- GLM

For actual plugin implementation:
- Windows 10
- Autodesk Maya 2018
- Visual Studio 2015

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

List the complete set of features to be included in the alpha version. For example, algorithmic functionality, GUI features, MEL or standalone application implementations, etc.
- Stand alone Qt application
- includes basic functionality for merging two meshes

Describe the important development milestones.
- Completed algorithmic work for preprocessing inputs
- Functioning mesh deformation and mesh optimization of merging for output

Describe the demo/test app you plan to create to show off the alpha features.
- Qt demo test app that will function as the Maya plugin is intended to function.

2.3.2 Beta Version Features

List the complete set of features to be included in the beta version.
- Same as Alpha but implemented in Maya
- UI scripted with Mel for Maya

Describe the important development milestones.
- Functioning UI in Maya.

Describe the demo/test app you plan to create to show off the beta features.
- Maya plugin demo.

2.3.3 Description of any demos or tutorials

How will users know how to use your tool? Describe the demos and/or tutorials you plan to develop that will ship with the final version.
- Video demonstration of how to use the tool with a few different mesh combinations
- A pdf of few different output pairings of novel object combinations
- The github repo will also contain a full development readme and possible use cases for this plugin.

3. WORK PLAN

3.1. Tasks and Subtasks

Task 1: Address design doc comments Duration: 8 days (spring break)
Task 2: Develop Test Framework / General OpenGL Implementation Duration: 16 days
Task 3: Additional Debugging (mandatory) and User Volumetric (stretch) Duration: 18 days
Task 4: Porting to Maya Duration: 17 days
Task 5: Final debugging and testing functionality Duration: 15 days
Task 6: Creating output meshes Duration: 6 days
Task 7: Creating presentation and final write-up Duration: 4 days

Task 1: Address design doc comments

- Subtask 1.1: See initial comments for design doc. spring break.

Task 2: Develop Test Framework / General OpenGL Implementation

- Subtask 2.1: Preprocessing input shapes
  - Annotate the shapes into semantically meaningful parts according to method described in [Kalogerakis et al. 2010]. *assigned to Hannah*
  - Set up way to properly compare these meaningful parts of each mesh between the two designated meshes. *assigned to Wenli*
- Subtask 2.2: Labeling for volumetric design restriction
  - For input geometry models, label each face with a restricted or free zone label. *assigned to Wenli*
- Subtask 2.3: Write necessary C++ classes
  - Write classes as described in section 2.1.3. *assigned to Both*
- Subtask 2.4: Implement Configuration Energy measure
  - The configuration energy will be computed based on a vector of weights based on a formulas from the paper. *assigned to Hannah*

Task 3: Additional Debugging (mandatory) and User Volumetric (stretch)
- Subtask 3.1: Buffer time for fixing additional bugs that may arise.
    - Fix bugs as needed since we will be using almost all the C++ code here directly as the backend basis of the Maya plugin. *assigned to Both*
    - Clean up the code and comment for readability. *assigned to Both*
- Subtask 3.2: User Volumetric Restrictions
    - To start, the algorithm will be relying solely on a paper implemented volumetric restriction for algorithm. User inputted restrictions might be added in later depending on if time allows. *assigned to Hannah*

Task 4: Porting to Maya
- Subtask 4.1: Preliminary Maya Integration
    - Compare mesh representations to confirm that the port from Qt to Maya can be done properly. *assigned to Both*
- Subtask 4.2: Copy Paste
    - Copy all C++ code for the algorithm from the Qt project setup into the Visual Studio development environment. Start porting over additional C++ to MEL interaction features to make the Maya plugin connected properly. *assigned to Hannah*
- Subtask 4.3: MEL User Interface
    - Setup MEL interface for user to interact with when operating the plugin. *assigned to Wenli*

Task 5: Final debugging and testing functionality
- Subtask 5.1: Debug Debug
    - Make sure all aspects are fully implemented properly. *assigned to Both*
    - Clean up the code and comment for readability. *assigned to Both*

Task 6: Creating output meshes
- Subtask 6.1: Meshes Galore!
    - Make fun output meshes to use in the final product write up. *assigned to Wenli*
    - Make a video demo of how to use the plugin and make fun output meshes. *assigned to Hannah*

Task 7: Creating presentation and final write-up
- Subtask 7.1: Presentation
    - Make the presentation and fill out all features of the readme on the github repo for others to read when interested in using this plugin. *assigned to Both*

3.2. Milestones

3.2.1 Alpha Version

Task 1: Address design doc comments

- Subtask 1.1: See initial comments for design doc. spring break.

Task 2: Develop Test Framework / General OpenGL Implementation

- Subtask 2.1: Preprocessing input shapes
- Subtask 2.2: Labeling for volumetric design restriction
- Subtask 2.2: Write necessary C++ classes
- Subtask 2.3: Implement Configuration Energy measure

Task 3: Additional Debugging (mandatory) and User Volumetric (stretch)

- Subtask 3.1: Buffer time for fixing additional bugs that may arise.
- Subtask 3.2: User Volumetric Restrictions

3.2.2 Beta Version

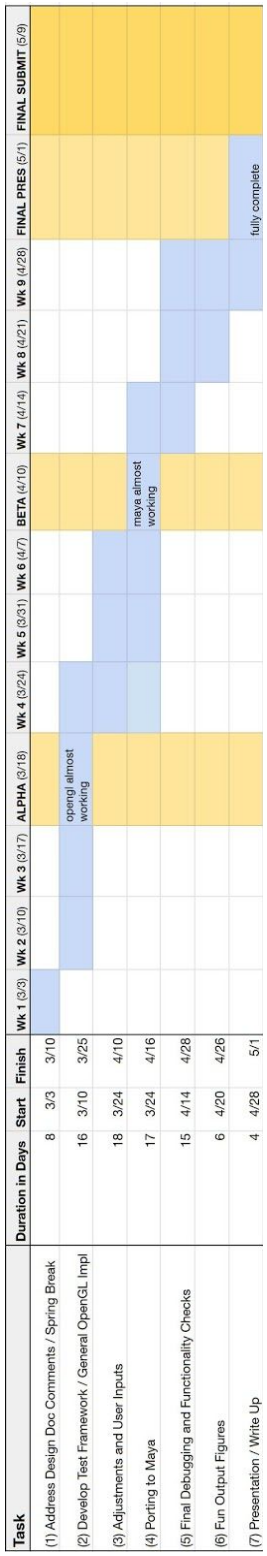
Task 4: Porting to Maya

- Subtask 4.1: Copy Paste
- Subtask 4.2: MEL User Interface

Task 5: Final debugging and testing functionality

- Subtask 5.1: Debug Debug

3.3. Schedule



[Schedule in the form of a gridded Gantt Chart]

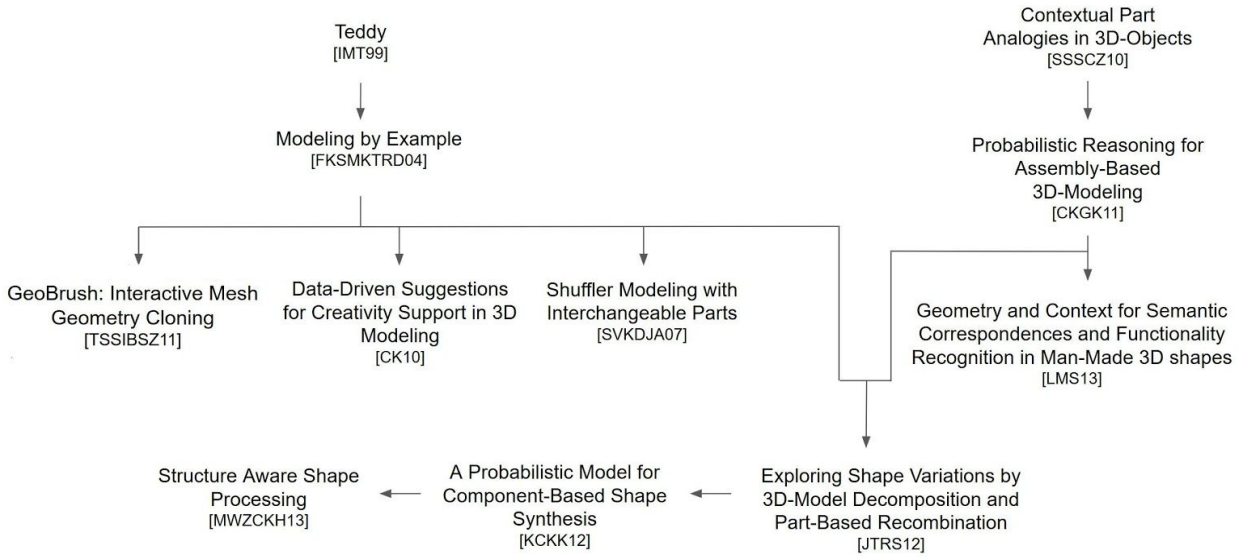
4. RELATED RESEARCH

Chimera, Literature Survey

Research Evolution

Tree of Life for Chimera Development

Research Evolution Graph for “Zoomorphic Design”



Noted Papers

[IMT99] Teddy: A Sketching Interface for 3D Freeform Design was a seminal paper that presented a system for quickly and easily designing freeform objects. Its major contributions included various construction algorithms that created 3D polygonal surfaces from a 2D silhouette. It was one of the first papers to present a semi-automatic method for creating models and led to more research on automatic and semi-automatic 3D model generation.

[FKSMKTRD04] Modeling by Example: A paper following Teddy by Funkhouser et al. presented a data-driven approach to constructing 3D geometric models by synthesizing existing models. The learning curve for creating 3D models is steep and the benefit of their approach was that it was easy to learn and create new models. The

users would search an existing database of 3D meshes, cut out the desired parts and composite them to form new 3D meshes. This paper contributed new research on interactive segmentation of 3D surfaces, shape-based search to find 3D models as well as composition of 3D parts. Their system allowed untrained users to create interesting and detailed 3D models. The paper learned from sketch modeling tools such as Teddy by keeping the user interface simple. Whereas previous systems achieved simplicity by limiting the complexity and shapes a user could create, their system achieved simplicity by leveraging complex existing geometry stored in a database. Their work was also the first to apply data-driven synthesis to 3D surface modeling such as identifying similar parts of 3D objects from a database and how to stitch them together. The system was also one of the first to leverage a large database of example 3D models and shape-based search methods as a part of an interactive modeling tool. In terms of Shape interpolation, the focus was more so on recombining parts of shapes rather than morphing between them. The model created a new paradigm for creating 3D models from parts extracted from a large database, as well as algorithms for segmenting models into parts interactively and aligning parts automatically.

[SVKDJA07] Shuffler Modeling with Interchangeable Parts: Building on top of the model creation from parts of other models paradigm of “Modeling by Example”, Sheffer et al. presented Shuffler, a modeling system that automates the process of creating new models by composing interchangeable parts from different models within each family. The parts that they automated was the geometric operations on the part of the user. The user would simply select which parts should come from which input models and the Shuffler system combines the chosen parts. The system precomputes interchangeable parts across each input family of models. To accomplish this, it segments the models into meaningful components and computes the correspondence between them. The paper presented two new novel algorithms for segmentation and part correspondence. This allowed for meaningful mesh segmentation and part matching.

[CK10] Data-driven suggestions for creativity support in 3D modeling: In this paper, Chaudhuri and Koltun propose data-driven suggestions for 3D modelling. Whereas previous semi-automated modeling approaches allowed for users to search themselves for model parts, this approach computes and presents components as suggestions to the artist. The paper builds of existing work for shape retrieval and shape correspondence techniques and use those techniques to support the generation of data-drive suggestions. The paper still uses a library of existing models, but automatically uses data-driven techniques to create suggestions. The benefit of suggestions over user-selection is that the models created can be more open-ended.

Whereas a user must know and decide what they want in earlier methods and systems, in this system, they are allowed to choose based on selections. This method is useful for conceptual phases of design and could help concept artists for films and games.

[SSSCZ10] Contextual part analogies in 3D objects: Coming from the Computer Vision side of research, Shapira et al. presented an approach for partitioning objects into meaningful parts and finding analogous parts in other objects. This paper is not building off of the semi-automated 3D modeling systems described above, but rather addressing a problem that is crucial to creating effective systems. In the approach to finding analogies between parts of 3D objects, they define a similarity measure between two parts based on local signatures and geometry as well as the context within the shape they belong to. The context is used in a part-in-whole matching based on a bipartite graph. The matching function is computed using a flow algorithm that takes into account both the local geometric features and the partitioning hierarchy. In addition to finding part analogies from shape repositories, they demonstrate an annotation tool that carries textual tags from object parts from one model to another.

[CKGK11] Probabilistic reasoning for assembly-based 3D modeling: At this point, the form of modeling in which new models are assembled from components extracted from a database was called “assembly-based modeling.” Earlier papers provided the foundation of assembly-based modeling. A challenge still was identifying relevant components for the user. This paper introduced probabilistic reasoning to assembly-based modeling. The model learns a probabilistic graph that encodes the semantic and geometric relationships among shape components in a given repository. The model increases the relevance of the components presented to the user.

[TSSIBSZ11] Geobrush: Interactive mesh geometry cloning: Building off of Modeling by Example, Takayama et al. proposed a method for interactively cloning 3D surface geometry with a paintbrush interface inspired by continuous cloning brushes used in image editing. They implemented a method that supports real-time continuous copying of arbitrary high-resolution surfaces. They used a novel method of generalized discrete exponential map parameterization to create a correspondence between source and target geometries. They align the source geometry with the target shape using Green Coordinates and compute an offset membrane to blend the pasted patch with C^1 continuity before stitching it to the target. They compute the offset membrane, which is a solution of a PDE, on the GPU in real time. Their method was the first to provide a complete solution that could handle general input for 3D surface cloning in real-time. The paint interface can handle irregular meshes and arbitrary surface details. GeoBrush

is significantly different from normal displacements and allows for more creative expression.

[KCKK12] A probabilistic model for component-based shape synthesis: The primary contribution of this paper is a new generative model of component-based shape structure. The model allows for more automation compared to previous systems. Their work focuses on designing a compact representation of the relationships of the selection and placement of components of complex real-world models from airplanes, cars, to furniture and biological forms. The model creates a compact representation that can be learned without supervision from a limited number of examples. Their probabilistic model of shape structure can be trained on segmented shapes from a particular domain. The model gives way to two applications: amplifying an existing shape database and enabling interactive shape synthesis interfaces for rapid creation of plausible shapes. The first application could synthesize new shapes and expand an existing database. The second application affords more user control similar to previous methods, but allows for more rapid creation.

[JTRS12] Exploring shape variations by 3d-model decomposition and part-based recombination: This paper takes us much closer to zoomorphic design. Jain et al. present a system in which new shapes are created by blending between shapes taken from a database. Rather than assembly-based modeling, they perform blending by recombining different parts from the two shapes using shape analysis and deducing the necessary constraints. The analysis includes shape segmentation, contact analysis and symmetry detection. The key simplification of this method is to avoid geometry inside of individual parts that constitute a shape. Similar to previous methods, they segment database models into parts and synthesis the new models from combining the component parts. In comparison to systems where users specify individual parts to be replaced, such as the Shuffler approach, their system performs complete blends. They also have a different segmentation and shape-matching strategy relying on positional and contact information instead. The approach does not allow for the deformation of individual parts, so shapes have puzzle-like structures.

[LMS13] Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes: This paper addresses the issue of balancing the automatic recognition of generic man-made 3D shapes while balanced with large topographical variations that make the more detailed man-made shapes harder to distinguish. To do this, they represent a graph of interspatial relationships between the connecting parts in the shape to model similarities. This allows piece-wise correspondence between underlying shapes of the larger object to be recognized

without needing user-specified denotations while also using design classifiers to learn the organization of the shape components of the current object for future uses of the algorithm. This context-aware measurement of the similarity between existing parts of the object is much more optimal than previous geometric-based techniques when dealing with larger shapes that have many more surface-based variations.

[MWZCKH13] Structure-aware shape processing: This paper focuses more on semantic relationships among different parts of an overall shape being created instead of the more detailed inter and intra relationships between the local geometries themselves. Using more basic 3D models as reference for look up, this algorithm analyzes then processes key extracted information for synthesizes towards the more novel shapes. Using a mixture of previously defined and available structure-aware techniques to combine larger databases of simple shapes into more specific creations. This is helpful because it removes the difficulty of the content creation bottle-neck and allows researchers and other tinkerers to focus more on the end product with faster and more iterable designs than original detailed 3D models that can take a longer time to create. For example, instead of constructing a new chair every time for iteration, the algorithm can pick different aspects of previously created chair designs to form a new, not yet seen combination (such as the recent StyleGAN by Nvidia that creates faces that don't exist from currently existing faces previously submitted to their database [KALL17]).

References

[CK10] CHAUDHURI, S., AND KOLTUN, V. 2010. Data-driven suggestions for creativity support in 3D modeling. ACM Trans. Graph. 29, 6 (Dec.), 183:1–183:10.

[CKGK11] CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3D modeling. In ACM Trans. Graph., vol. 30, 35.

[DYYT17] DUNCAN, N., YU, L.-F., YEUNG, S.-K., AND TERZOPOULOS, D. 2015. Zoomorphic design. ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH, 34(4):95:1-95:13.

[FKSMKTRD04] FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. In ACM Transactions on Graphics (TOG), vol. 23, ACM, 652–663.

[IMT99] IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3D freeform design. In ACM Trans. Graph., ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 409-416.

[SVKDJA07] SHEFFER, V. K. D. J. A. 2007. Shuffler: Modeling with interchangeable parts. Visual Computer journal.

[JTRS12] JAIN, A., THORMAHLEN ¨ , T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Exploring shape variations by 3d-model decomposition and part-based recombination. In Computer Graphics Forum, vol. 31, Wiley Online Library, 631–640.

[KALL17] T., KARRAS, T., AILA, S., LAINE, AND J., LEHTINEN. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. Article Karras2017ProgressiveGO, CoRR 1710.10196.

[KCKK12] KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. ACM Trans. Graph. 31, 4, 55.

[LMS13] LAGA, H., MORTARA, M., AND SPAGNUOLO, M. 2013. Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes. ACM Trans. Graph. 32, 5, 150.

[MWZCKH13] MITRA, N., WAND, M., ZHANG, H. R., COHEN-OR, D., KIM, V., AND HUANG, Q.-X. 2013. Structure-aware shape processing. In SIGGRAPH Asia 2013 Courses, ACM, 1

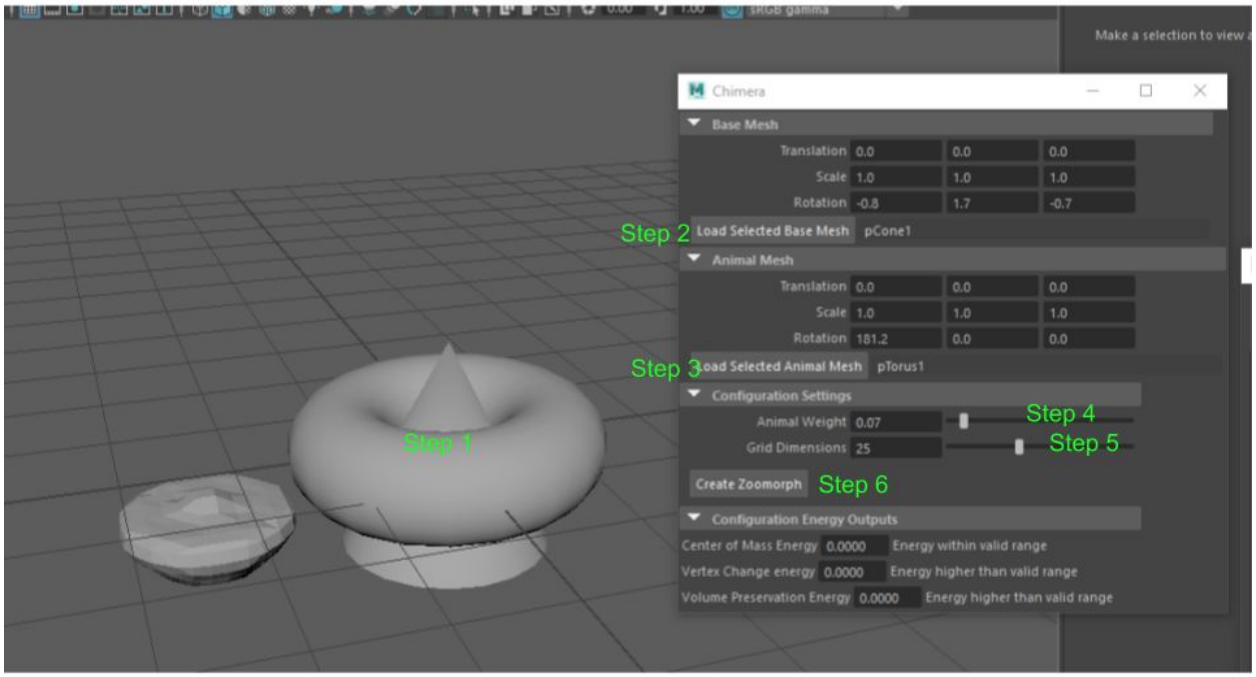
[SSSCZ10] SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. 2010. Contextual part analogies in 3D objects. International Journal of Computer Vision 89, 2-3, 309–326.

[TSSIBSZ11] TAKAYAMA, K., SCHMIDT, R., SINGH, K., IGARASHI, T., BOUBEKEUR, T., AND SORKINE, O. 2011. Geobrush: Interactive mesh geometry cloning. Computer Graphics Forum 30, 2, 613–622.

5. Final Report Review

Results

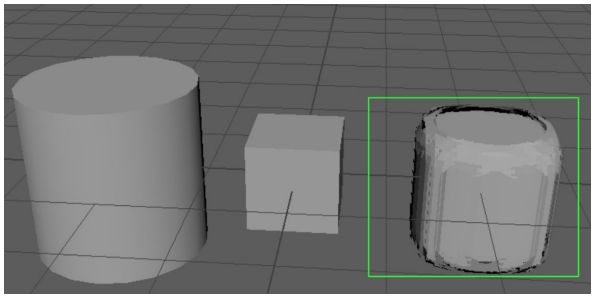
- Documentation / Tutorial
    - Step 1: Import a base mesh and animal mesh into the current Maya scene.
    - Step 2: Select the base mesh and click the “Load Selected Base Mesh Button”
    - Step 3: Select the animal mesh and click the “Load Selected Animal Mesh Button”
    - Step 4: Adjust the slider for “Animal Weight”
    - Step 5: Adjust the slider for “Grid Dimensions”
    - Step 6: Press the “Create Zoomorph” button to create a new mesh.
    - (Optional) Step 7: Delete the new mesh and start over.
- Content Creation

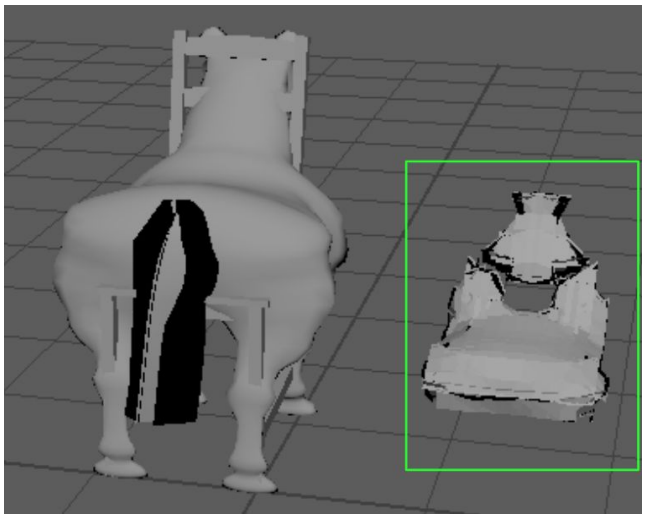


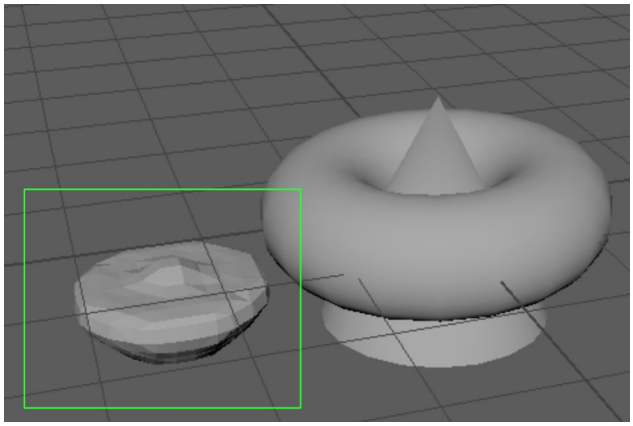
1. The user can use any meshes that can be loaded into the Maya interface.
2. After selecting an object and pressing “Load Selected Base Mesh”, the appropriate mesh node name is stored in the Maya plugin and will be passed to the plugin backend.
3. Similarly, the animal mesh will be stored

4. The animal weight slider is a weighting that determines how much influence the animal mesh will have on the final output mesh. The higher the weighting, the more influence the animal mesh will have. At a weight of 0, the final mesh will be based entirely on the base mesh.
5. The grid dimensions determine the level of detail on the final output. The smaller the grid dimension, the faster the output can be generated, but the more grid-artifacts will be apparent in the output mesh.
6. The node creates a single output mesh based on the two inputs.

Content Creation Examples:

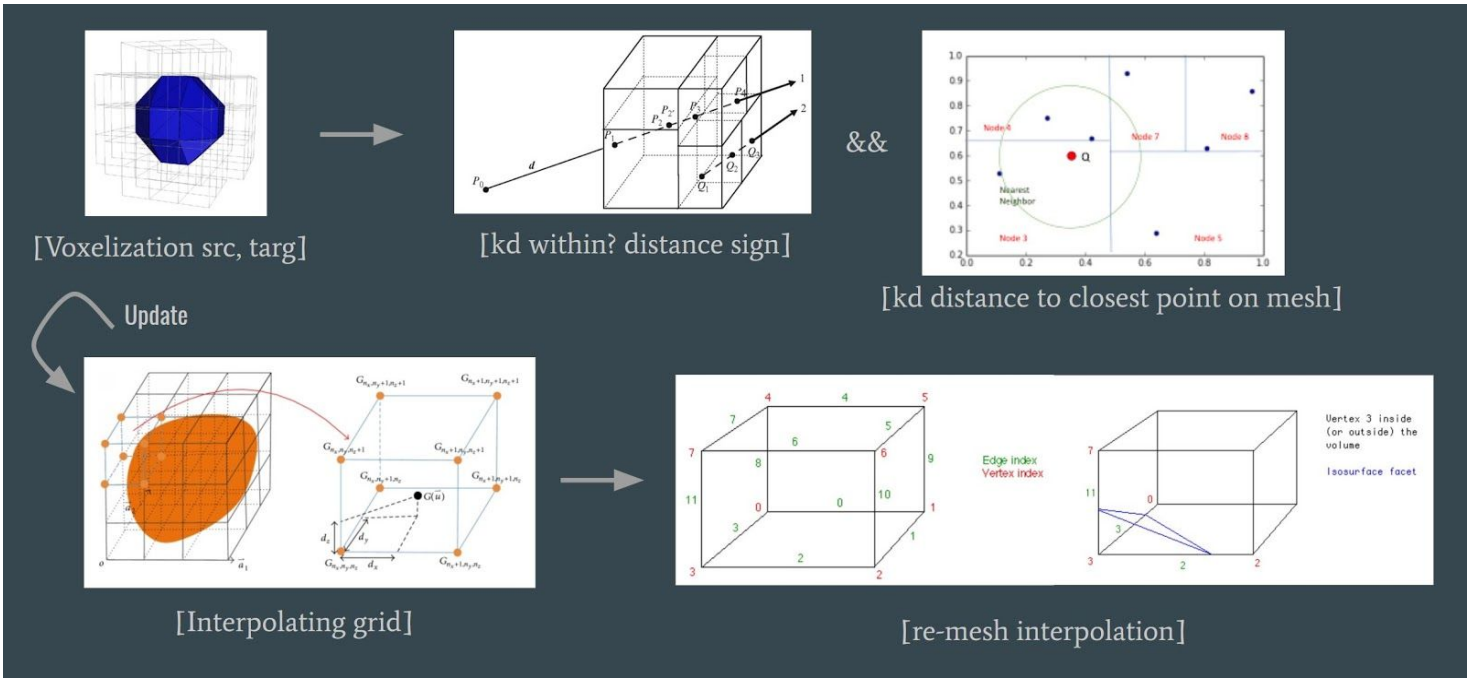






Discussion

- Accomplishments



  - Features implemented:
    - Mesh interpolation
      - K-d tree acceleration
      - Marching cubes-based remeshing
      - Signed distance calculations
    - We ultimately used a mesh interpolation method based on level sets which led to an implementation of a K-d tree acceleration structure for the meshes and calculations of signed distance functions for each mesh that allowed us to interpolate the two meshes. Based on the level set of the base mesh and animal mesh, we could calculate a grid that represented whether a point was inside the target mesh at any point along the interpolation. Using the grid and a marching cubes-based re-meshing technique, we created a new mesh based on the desired weight of interpolation.
    - We stepped away from the methods described in our original design document. For more discussion on why see the Design Changes section below.
  - Features not implemented:

        - Because we kept refactoring our implementation, volumetric design restrictions were not trivial to implement, so we left them to the end in case we had time to do so.
- Design Changes
    - We refactored our entire project from the original proposed implementation to one that we worked out ourselves. To start, the original paper was a thesis that pulled on about four different other seminal papers that would have also acted as full plugin projects themselves. We tried re-scoping our project to include fewer of the original aspects to still follow our original paper; however, all the main papers that it drew on were intrinsically required by each other for our original paper to function as expected. The main issue was due to correspondence matching between the two shapes to properly interpolate between them. One way to resolve this would have been to do mesh painting between the two, yet after lengthy discussions we ultimately determined that this would have hindered the usability of our plugin and opted not to implement this. Instead we worked through a couple different paper implementations until we found one that interpolated nicely with a re-forming mesh and could be mathematically solved in almost real-time by the user (to again help with usability for our plugin). All-in-all, we switched from the animal to object interpolator to an object to object interpolator that works using level-sets and marching cubes.
- Total Man-Hours Worked
    - 61.0 hours per person
- Future Work
    - volumetric design restriction - Adding in a feature for skewing the interpolation for some of the level set more towards one of the meshes in comparison to the other at that point in the level set (a way to do a face+cup combination instead of just interpolate between the two of them).
    - Now that we know the implementation with the level set is very parallelizable, a next step for making this actually real-time would be to introduce GPU optimizations allowing for a much more fine-tuned grid and still relatively good output with it in comparison (the more detailed the grid, the better the result, but right now it's full CPU so the more detailed the grid the much slower the runtime).
- Third-Party Software
    - Qt for debug setup and Visual Studio for setting up the backend of the plugin
    - glm for handling the linear algebra and vector aspects of our manipulations
    - we used eigen and glm at the beginning but after changing how our project was implemented, glm as the sole backend for the linear algebra worked better

- Additional Source material - since we switched to our own basis of implementation, here's the resources we used for it:
  - "A Level-Set Approach for the Metamorphosis of Solid Models"
    David E. Breen, Ross T. Whitaker, California Institute of Technology, School of Computing University of Utah Salt Lake City
    IEEE Transactions on Visualization and Computer Graphics, 2001
  - Dynamic Occlusion With Signed Distance Fields
    Daniel Wright, Epic Games
    SIGGRAPH, 2015
  - "Generating Signed Distance Fields From Triangle Meshes"
    J. Andreas Baerentzen, Henrik Aanaes, University of Denmark
    IMM-TECHNICAL REPORT, 2002
  - "Isosurfaces and Level-Sets"
    Ross T Whitaker, School of Computing University of Utah Salt Lake City
    The Visualization Handbook, 2004
  - "Level Sets and Fast Marching Method"
    Gilbert Strang, MIT
    (part of his textbook), 2006

Lessons Learned

- Don't look at parts of a paper as black boxes. When a paper author briefly describes a method, do not assume it is simple because they did not devote a large part of the paper explaining it, and don't assume that you'll be able to get working code from them if they say it's available. Do more research into how much technical work a paper is built upon.
- Write a script in MEL and add it to your Maya shelf to load and unload your plugin quickly as soon as you get started. Anything that can shorten your iteration and build cycle can greatly increase the speed of your development workflow.
- Sadly we did not have the opportunity to do this due to our refactoring; however, it would really help if students start the Maya integration as early as possible. We started doing this alongside our Qt integration, building both at the same time.
- Working on extensive research projects can be rewarding and frustrating at the same time when trying to work with other individual's projects (ie when trying to replicate their work)