Maya Paint and Retopology Pipeline Tool

By

Nina Olga Kokot

MSc Computer Animation and Visual Effects
National Centre for Computer Animation
Department of Media and Communication
Bournemouth University, Bournemouth, UK

August 2024

Supervisor Jonathan Macey

ABSTRACT

Topology is a geometric characteristic of 3D mesh represented by a number of faces and vertices creating a layout in a three-dimensional space. It is an important element of the pipeline, but its importance is determined mostly by the fact that good topology influences rendering time, the process of rigging, which is a process of preparing the asset to be moved, and animating, the model. Usually, before the 3D mesh can be used in the next stages of production it needs to be re-topologised, which is a process of re-creating the original mesh or altering it. There are various tools available for re-topology that vary in approaches. One of the most popular ones is Quad Draw Tool from Autodesk Maya. It is a simple but powerful solution that allows the artists to prepare their models for production. This method however is time-consuming and not very intuitive for beginners. The information on how to make a good topology is limited online. Pixar Animation Studios is one of the leading industry studios that published papers and materials on topology and how to prepare your mesh to be fit for the production pipeline. However, it is not all in one place and one must do thorough research to be confident in the re-topologising skills. This project aims to create a plugin that eases the user into the re-topology workflow and provide examples and guidance throughout. For more experienced artists it is a tool that can help speed up a process and therefore improve the pipeline. It is based on the two tools available in Maya, Paint Vertex Color and Quad Draw. The idea is to simplify the process of activating both tools and following the usual necessary steps to create a new mesh. It combines many steps into one or two buttons instead. The current workflow can also be confusing, especially if working on big meshes. This tool can be used as an alternative to the current re-topology workflow that many follow and instead help speed up a tedious and prone-to-mistakes process.

Key words: *brush, GUI, Maya, paint, Pixar, re-topology, topology, user interface, vertex*

DEDICATION

Dedicated to my family and friends who supported me throughout this project.

ACKNOWLEDGMENTS

I wanted to express gratitude to my supervisor for his guidance and to my friends for their patience and support during this project.

Contents

Chapter 1

Introduction

1.1 Introduction

Topology is a fundamental part of 3D modelling. It refers to creating a geometric mesh that will later be used in the production pipeline, so it must follow some standards commonly used by 3D modellers. Pipeline is an industry term for a series of different stages in creating 3D or CGI productions.

The importance of topology is determined by a wide range of reasons, such as UV unwrapping, texturing, rendering time and memory saving, but mostly by how much rigging and animating depends on it. One of the essential parts of creating a topology suitable for those stages of the pipeline is ensuring that the main areas of deformation of the mesh are clearly determined.

Often the initial models made by 3D artists are not suitable for production until they undergo a process called re-topology. That method is either re-making a new mesh on top of the original one or optimising the current model.

Nowadays, there are many different tools available for re-topology, each DCC (Digital Content Creation) has its own way of doing it that varies in implementations and uses. The one that is widely used is from Autodesk Maya and is called Quad Draw Tool. It is based on the idea of drawing vertices and connecting them on top of the original mesh. The current workflow for re-topology available in Maya is not very efficient nor easy for new artists to navigate.

1.1.1 Aims

The aim of this project is to create a tool that helps with introducing the user to the 'painting on the mesh first' re-topology workflow and that provides them with different options for the artists to choose from that will fit their approach best. One of the main goals for this project is to design a well-thought interface for the user. The key challenge is to make the final tool useful and easy to navigate. Another aim is to limit the amount of unwanted mistakes that are often being done during re-topology and reduce the time needed to be spent on this process.

1.1.2 Objectives

In order to make this new tool it is crucial to use the right resources available. Autodesk Maya has already two tools that can be used in the proposed workflow of this tool.

Combining the Paint Vertex Color and Quad Draw tool from Maya will help automating the whole process of re-topology.

Creating the desirable interface is possible by using Pyside2 and Qt. By conducting a through research and providing the user with some references this tool should become an educational tool to some etent.

1.1.3 Thesis overview

Thesis overview Part 1. starting with this Chapter 1. contains Introduction, Aims, Objectives and Thesis overview. Part 2. focuses on previous work in Chapter 2. In Chapter 3. the technical background is being presented. Part 3. is an implementation of Part 2. in Chapter 4. Part 4. shows results of Part 3. in Chapter 5. and concludes them in Chapter 6.

Chapter 2

Previous work

2.1 Related work

User experience is the main focus of this project alongside the goal of creating a useful pipeline tool that will improve the experience and reduce the time needed to be spent. The final output should help with the learning curve for novices and speed up the re-topology workflow for the other artists.

Pixar Animation Studios is a studio whose computer graphics research for years has been influencing the field and the production pipeline. Great example for this statement is a ground-breaking paper from 1998 "Subdivision Surfaces in Character Animation" by Tony DeRose et al. It introduces the subdivision surfaces into the pipeline and encourages it to be more wide-spread than it used to be. Instead of using NURBS (Non-uniform rational basis spline) surfaces, which are simple geometrical representations of curves, that offer topological restrictions, Pixar's researchers developed a method that relies on their own algorithm which allows creating smooth surfaces on the model. This method involves constructing a surface from a regular polyhedron and dividing each face many times to achieve a smooth surface (DeRose et al. 1998). The implementation presented in this paper relies on Catmull-Clark surfaces subdivision that are used to create a smooth surface. As mentioned in the paper, the equation underneath shows how to calculate new edge points as an average of edge end points and adjacent face points, where e_j^{i+1} is computed as shown in Figure 2.1

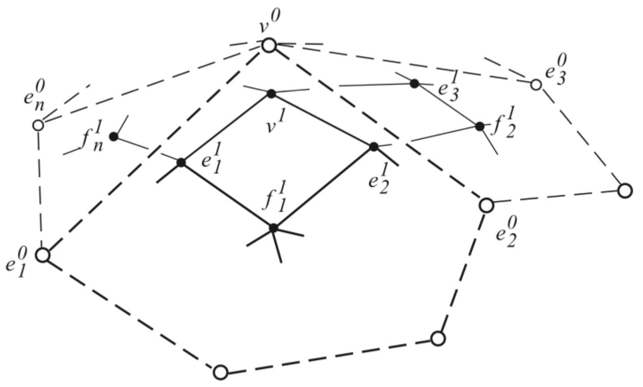


Figure 2.1: The case surrounding the v^0 vertex of valence n (DeRose et al. 1998,p.3)

$$e_j^{i+1} = \frac{v^i + e_j^i + f_{j-1}^{i+1} + f_j^{i+1}}{4} \tag{2.1}$$

Then eventually the new position of the vertex point v^i is calculated as

$$v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e_j^i + \frac{1}{n^2} \sum_j f_j^{i+1} \tag{2.2}$$

Pixar's researchers adapted this method from Catmull-Clark subdivision surface to their own needs and integrated it into RenderMan, Pixar's rendering software, which required some changes and additional math solutions but in the end they achieved the desired goal and had great results.

Introducing subdivision surface technique into pipeline allowed the modellers to arrange control points of the model into more naturally looking geometry that helped with achieving the final look of a real-life characters (DeRose et al. 1998). In the end, their new technique proved to be so useful that most of the DCCs nowadays have their own surface subdivisions method, including Maya, and it forever changed the modelling process and the approach to the topology.

In Course Notes "Art and Technology at Pixar, from Toy Story to today" made for SIGGRAPH ASIA 2015 by Pixar Animation Studios presenters, Villemin R., Hery C., Konishi S., Tejima T., Yu D., studio's modeling pipeline is described in more detail. It starts with a 3D scan of a digital mesh or of character's real-life sculpt. That scan is then adjusted for rigging by correcting the topology and subdiving it. In those course notes, it is also mentioned how they improved their cloth pipeline. Before, creating "clothing followed a 2D pattern tailoring system" (Villemin R., et al. 2015, p. 8), which meant that only people who had enough pattern drafting knowledge were able to use it. Now, more modelers can be involved in this part of a pipeline and therefore improve it. Pixar put great focus on refining that tool to accommodate artists without the more-in depth knowledge.

"Sketch-based modeling: A Survey" by L. Olsen, F. Samavati, M. Sousa and J. Jorge is a great example of a similar approach in relation to GUI to the one proposed for this plug-in. This paper considers how the classical user interfaces for modelling tools can be discouraging and over-complicated or non-intuitive for novices. In their tool, the authors proposed a more accessible way of creating an interface that was more "artist-friendly". Based on the idea of allowing the user to sketch in the tool, the user interface for modelling for that tool is supposed to be sketch-based (SBIM). The goal of this paper was to explore options of mapping 2D sketches into 3D models and making the GUI (Graphical User Interface) easier to navigate for less technical artists. In this case, the input is mostly acquired from pen-based or other devices with screen display. Later, the sketches are embedded into 3D by projecting onto drawing canvas or mesh itself (Olsen et al. 2009). SBIM (Sketch-based Interfaces and Modeling) is the authors solution to the traditional windows-based interface that proved to be not useful for their approach. "Unlike a command selected from a menu, freehand input is inherently ambiguous and open to multiple interpretations." (Olsen et all. 2009, p.90) This means that this GUI is supposed to offer many different options depending on what is the artist's intent, if they want to create fully 3D model from their sketch or just deform the existing one with their sketches. For the interface design, paper focuses on creating a final product that is intuitive and easy to navigate even for new modellers without a need of learning new and complicated tools. A common approach in sketch-based systems is supporting gestures that resemble real-life sketching process. "Some examples of gestural commands are cutting and deleting strokes, object grouping, erasing and local smoothing , and stroke blending" (Olsen et al. 2009, p. 96). In the paper, it is also mentioned that even with a different approach to creating a new interface, the GUI might be less overwhelming for beginners but it still requires some getting used to and therefore clear instructions might be one of the ways to improve the user's experience.

2.2 Previous work

The final plug-in depends on Paint Vertex Color and QuadDraw tool that are Maya's internal functions. Achieving the correct topology at the end is one of the purposes of this project and by combining those two tools there is a new one created that has the functionality that QuadDraw tool does not has on its own. There are some examples in the industry of tools having similar approaches to the one proposed in this project, for many the most related would be Topogun and QuadPatch. It is also worth mentioning OpenSubdiv.

2.2.1 OpenSubdiv

OpenSubdiv is a set of open source libraries which were developed by Pixar Animation Studios. This tool is used mostly for rendering but it relies on Catmull-Clark subdivision surface algorithm that was improved to fit their own product better. It is used in many DCCs. It allows working with dense meshes and updating it in real time which proves useful since it is important to observe how changes are affecting the topology.

2.2.2 Topogun

Topogun is an application used for re-topology and baking maps. It has set of different tools for making new topology that allow the user to edit or draw directly on the mesh. It also has an automatic re-topology feature. Topogun also puts emphasises on a user-friendly interface that it is straightforward and can be easily personalised.

2.2.3 QuadPatch

QuadPatch is a great example of a re-topologising tool with a well-made interface that is easy to learn with enough descriptions and useful features, such as Dynamic Path Drawing, to allow the user to ease into re-topology workflow. It is a tool developed for Blender 3.0 that helps to save time when working on larger meshes without many details.

Chapter 3

Technical Background Research

3.1 Maya API

Maya API (Application Programming Interface) is an interface that allows programmers to access Maya's internal libraries in order to create new plug-ins, scripts, tools and nodes. It is available in C++, Python and MEL (Maya Embedded Language). In the context of this project, using Maya API is crucial because it helps automatise and systematize all the essential processes of creating a plug-in. After initial research, Python proved to be the best language to use for many reasons, including having many resources available, such as official Maya documentation, and generating less code than C++. For this project's purposes, 'OpenMaya' and 'maya.cmds' were considered and both were used in appropriate places.

3.1.1 OpenMaya

Initially, options of mostly using 'OpenMaya' module were considered because it allows direct geometry manipulation through different function sets, such as MFnMesh. After exploring different approaches with MPxNode, MPxDeformer and MPxToolCommand, MPxCommand proved to be the best choice because it has all the functionality required by Maya to execute the tool as if it were embedded in.

3.1.2 maya.cmds

'maya.cmds' is a Maya Python module that is a part of Maya API. It allows the programmer to access the available tools and commands of Maya in an accessible and easy-to-navigate way. It is also a wrapper for MEL which is a native Maya scripting language. This means it has thorough documentation that corresponds to the MEL's and therefore proved to be more useful for this particular project that required the use of many MEL commands as 'maya.cmds'. Using this module over 'OpenMaya' was also determined by the fact that the latter offers low-level access to Maya's internal interface while 'maya.cmds' is great for higher-level access to Maya functions.

3.2 Paint Vertex Color Tool

Paint Vertex Color is a tool from Maya, which allows the artists to assign colors directly to vertices of the geometry they are working on. It is often used instead of texturing or to create color masks for shaders for later. Some artists use it in the process of re-topology but it is not commonly used for this reason. This tool is an important part of the final plug-in as its main idea is developed around it.

3.3 Quad Draw Tool

QuadDraw tool is one of the most well-known tools for retopology in the industry, as many studios use Maya and this tool is a part of it. QuadDraw allows the user to draw new faces directly on the 3D model. Its name comes from quad which means that is the point of the tool, first artist draws four vertices and then it creates a new polygon on the surface of the original mesh.

3.4 Qt

Qt is a framework for developing cross-platform applications which create graphical user interfaces. It is often used in the process of creating a new tool in Maya to customise its graphical interface. Qt's libraries are shipped with Maya and for the plug-in aspect there is Maya's own application object that can be passed into it therefore it does not require creating QtApplication at all.

3.4.1 Pyside2

Pyside2 is the official Python binding for Qt on Python. It comes pre-installed in Maya so it is more suitable for a plug-in developed for this particular DCC rather than using PyQt5 which would require additional installation. This would add an additional step in setting up the tool because the purpose of this project is to make it easy for beginners and since essentially both PySide2 and PyQt5 are using the same libraries it is not needed.

3.5 Topology research

In this section, the topology research was conducted and its results are shown. According to Pixar, well-constructed subdivision geometry have those key characteristics in common:

1. "They consist primarily of regular faces (quads for Catmull-Clark, tris for Loop)
2. They contain few extraordinary vertices

3. They efficiently describe the intended shape

4. They are topologically manifold "

(Pixar, 2024)

Two more important techniques that are also mentioned in 'Modeling Tips' on Pixar's website are Edge-Loop Transitions and Triangles and N-Gons.

The first one is based on the fact that sometimes in order to control the density of the mesh, it is necessary to use "extraordinary vertices" that can help with the amount of edge loops and flow of the topology. It is often needed in the areas around fingers, on the head or on the back where the deformations are not happening directly but it is still important to maintain those areas. In the underneath example this method is shown with using a valence 5 vertex to lower 2 edge loops into 1 (Fig. 3.1).

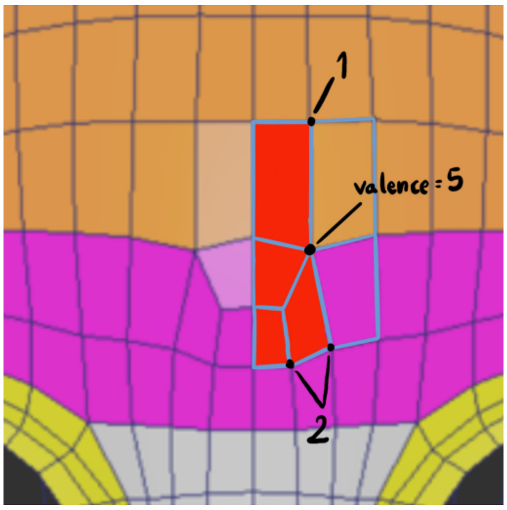


Figure 3.1: An example using a valence 5 vertex to lower 2 edge loops into 1 (original

The Triangles and N-Gons technique is about using the non-quadratic polygons in the topology in the strongly deforming areas such as shoulders, arm pits or hips. It is not a good practice to use many of triangles or polygons with more than four faces but it is allowed in those areas as long as it helps with maintaining a smooth surface and right topology flow (Fig. 3.2).

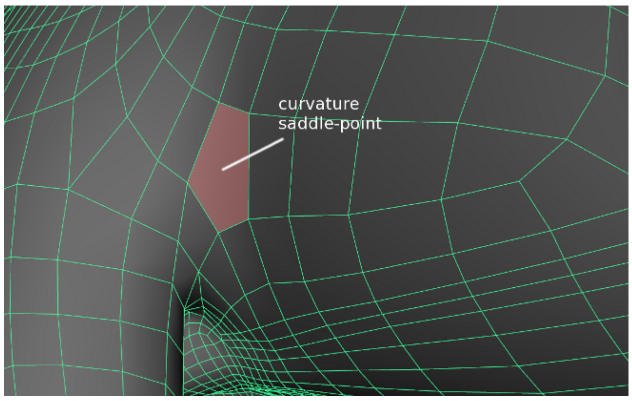


Figure 3.2: An example of using N-Gons technique (Pixar, 2024)

Chapter 4

Implementation

4.1 Overview

In this chapter, the implementation of conducted research and the final solution is presented. The final product's goals are to have an interface that is easy to navigate for novices and allows them to learn the techniques and to be useful and improve the pipeline for more experienced artists that will cut the unnecessary mistakes.

4.2 Initial tests

Initially, using only OpenMaya was considered but it proved to be too in depth for the amount of time given for this project to explore it enough. However, some research was conducted for it and it resulted in a test of writing a paint tool as a MPxDeformer (see Appendix 1). This approach proved to be over-complicated and not necessary when another test with maya.cmds was done. This new method was chosen over only using OpenMaya because it works on higher-level and therefore it saves time and allows access to tools such as Paint Vertex Color and QuadDraw which were essential for this plug-in.

4.3 Final implementation

The final *Maya Paint and Retopology Tool* plug-in was based on the technical background and created with the main goals of this project.

4.3.1 References

Using topology research, there were several references created to be later used in a tool as Image Planes that user can easily create and have on the side while working. It focused on the most important parts where deformations commonly occur, such as head, hands, knees, elbows and chest.

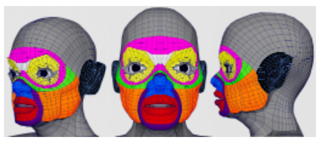


Figure 4.1: Stylised head reference

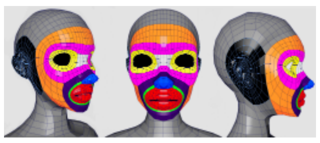


Figure 4.2: Normal head refernce

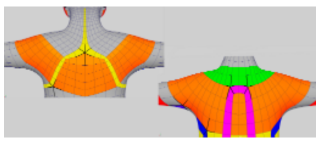


Figure 4.3: Back reference

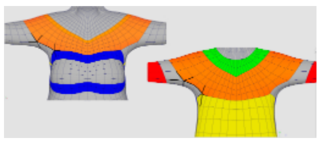


Figure 4.4: Front reference

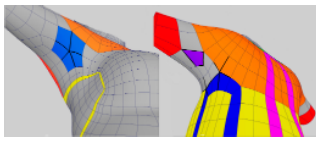


Figure 4.5: Arm/shoulder reference

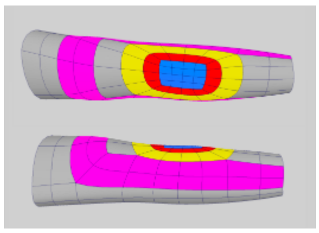


Figure 4.6: Elbow reference

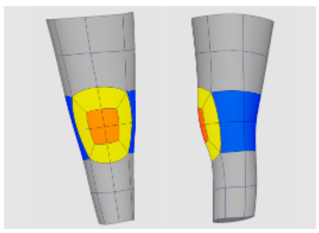


Figure 4.7: Knee reference

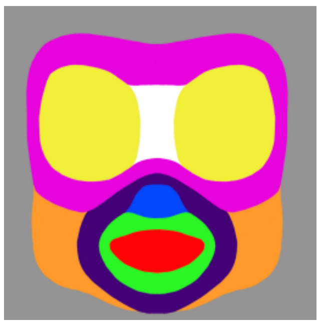


Figure 4.8: Template 1

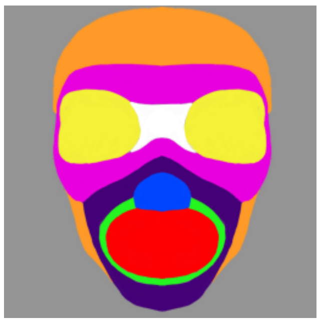


Figure 4.9: Template 2

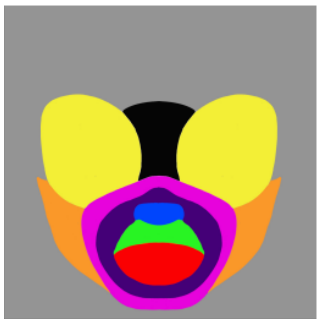
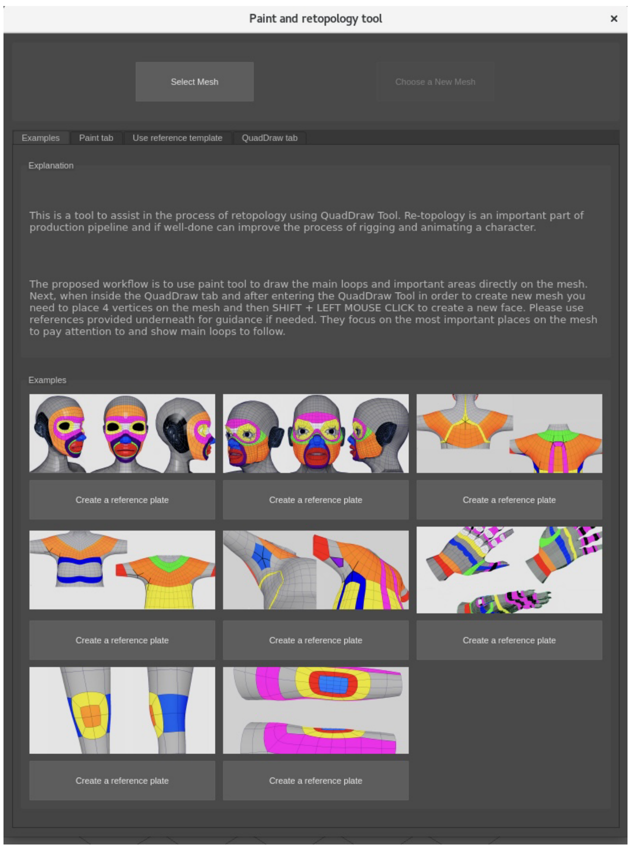


Figure 4.10: Template 3

All of the references and templates used in this tool were made specially for this project and are original.

4.3.2 Interface design

While developing the tool, the interface design was constantly being updated according to any changes and new features to ensure that it will be easy to navigate and accessible. For the design, it follows a straightforward format with two buttons at the top and four, clearly named tabs with different functions (Fig. 4.1 1). In order to make the GUI more user-friendly, there is a few hidden functionalities that are disabling or hiding different parts of the interface, depending on the user input. Initially, everything except for *'Select Mesh'* and create references buttons are disabled because it prompts the user to first select the geometry to work on and this is a crucial step for the tool to work. Another deactivation takes place when the user chooses to use a template on the model because unless they want to use the paint tool to fix the sides, it is not necessary for them to use it. It also hides the *'Focus only on one color'* button since it enables the option to work on one of the predefined colors from Paint tab. Reference template tab's functions change texture into color but the vertices are not being assigned any value so that option will not work when the template is used instead of the method of simply painting on the mesh. As mentioned before, there are four main tabs: *Examples*, *Paint*, *Use reference template* and *QuadDraw.* Each of those has different functionalities but they all depend on others in one way or another.



This is a tool to assist in the process of retopology using QuadDraw Tool. Re-topology is an important part of production pipeline and if well-done can improve the process of rigging and animating a character.

The proposed workflow is to use paint tool to draw the main loops and important areas directly on the mesh. Next, when inside the QuadDraw tab and after entering the QuadDraw Tool in order to create new mesh you need to place 4 vertices on the mesh and then SHIFT + LEFT MOUSE CLICK to create a new face. Please use references provided underneath for guidance if needed. They focus on the most important places on the mesh to pay attention to and show main loops to follow.

Figure 4.11: Design of the *Example tab*

- **Example tab**

  Example tab has two sections vertically placed: Explanation and Examples. The first one is using a QLabel and is only used as an introduction to the tool and quick explanation of the re-topology workflow proposed for this tool.

  Examples section uses a grid layout that is used to place the reference pictures in the right places with corresponding buttons underneath each of them. *'Create a reference button'* is using a command that creates and imagePlane in Maya with a right file path (Fig. 4.1 1).

- **Paint tab**

Paint tab is split into two sections as well: Brush size and Choose color. The former offers the user to change the size of the paint brush they are using with a slider and to activate or deactivate the symmetry according to different axis. Slider is created with *createSlider* function that calls two functions *changedValue*, which changes string put in the value box into a float, and *synValues*, that then takes that float and assigns it to the slider value that is connected to the brush size value. Symmetry buttons are simply calling an original options from Paint Vertex Color Tool.

Choose color group uses a grid layout as well because it is the most optimal way to display the colors in the desired way. Each paint button has different predefined color assigned to it. There are only nine colours available to the user as opposed to the original Paint Vertex Color because QuadDraw tab functions depend on it and also because it is to make the user to follow the provided examples that are using the same limited colors (Fig. 4.12). There is also an *'Erase paint'* button for erasing the paint and *'Exit the Paint Tool'* button to come back to the Selection mode.

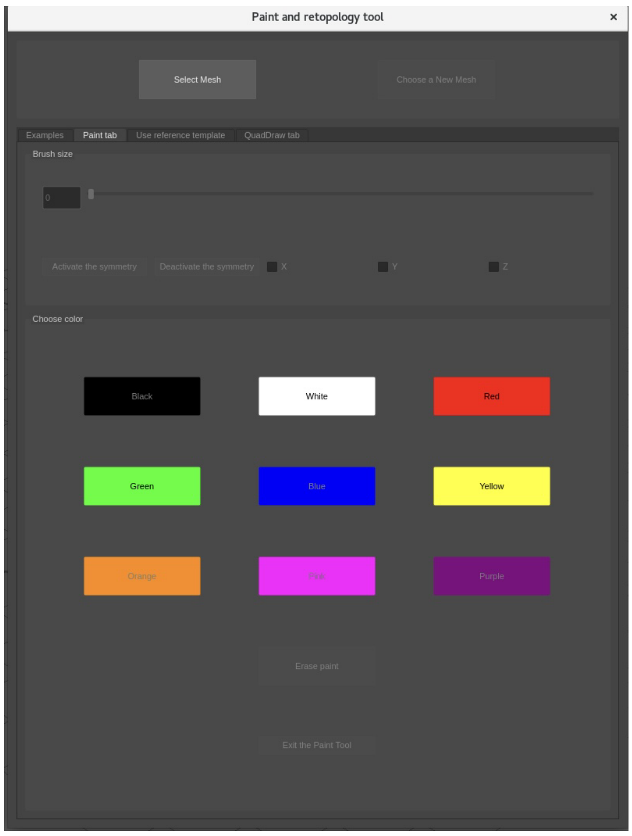


Figure 4.12: Paint tab

- **Use reference template tab**

  Use reference template tab uses grid layout in a similar way that Examples section from Examples tab does to display the available templates to the user and have matching button underneath. *'Use this'* button calls on *shaderCreator* function and disables Paint tab and hides *'Focus on one color'* button with all its option from QuadDraw tab.

Algorithm 1 shaderCreator

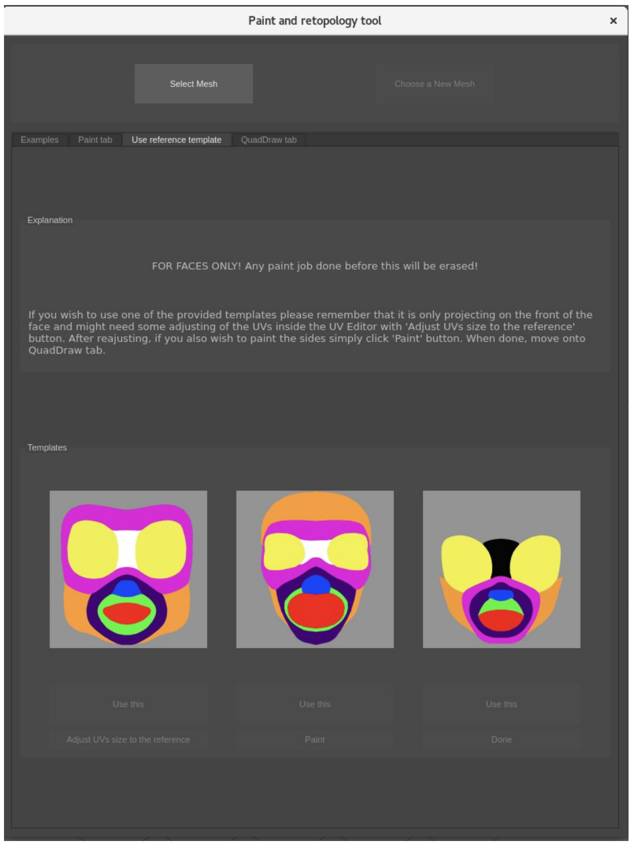
```
1: Set orthographic view and select front camera
2: Open UV Editor
3: Select mesh
4: Delete any UVs and then create new one that is camera-based
5: Toggle the UV texture image display option
6: Create shader with a chosen template
7: Create lambert shader called myShader
8: Create file texture node called myFile
9: Create place2dTexture called myPlace2dTexture
10: Connect place2dTexture to file texture node
11: Connect file texture node with shader
12: Set path to the file texture node with the chosen template
13: Create shaderGroup called myShaderGroup
14: Assign shaderGroup to the shader
15: Assign shader to the selected mesh
16: Change texture into color information
```

This option is meant for faces, not the rest of the body and offers a few different size options and painted mesh divisions.

'Use UVs size to the reference' button selects the UVs for the user who can then adjust them to fit their model best. Since this is the model that will be disregarded, their UVs do not need to be correct and can be used in this way to make them match the templates the best way possible.

'Paint' button enables Paint tab again and takes user straight into it to allow them to make any necessary changes on the mesh since the texture only works for the front of the face, but not the sides so if the user wants to improve the sides as well they can

do it with this button. 'Done' button exits the tool and comes back to the Selection mode.



FOR FACES ONLY! Any paint job done before this will be erased!

If you wish to use one of the provided templates please remember that it is only projecting on the front of the face and might need some adjusting of the UVs inside the UV Editor with 'Adjust UVs size to the reference' button. After reajusting, if you also wish to paint the sides simply click 'Paint' button. When done, move onto QuadDraw tab.

Figure 4.13: Use reference template

- **QuadDraw tab**

  QuadDraw tab has two sections: QuadDraw tool and Reminder. The first one uses vertical layout but when 'Focus on only one color' button is clicked that opens a group inside of it with color buttons using grid layout, similar to the Paint tab layout. *'START'* button calls the QuadDraw tool. In order to use this tool, the user would have to first make the object live and it is also a common practice to duplicate your original mesh and hide it to have it as a cautionary measure. This whole process of creating a duplicate, hiding the original and making the object live is done for the user instead when the start button is clicked. *duplicateMesh* function is responsible for duplicating the geometry and putting the original one on a new displayLayer that makes it invisible and unselectable.

'Focus on only one color' is enabled after *'START'* button is clicked and disabled. When this button is triggered it shows the same color buttons with the same color values assigned to them as in the Paint tab but are used differently. They all call *quadToolColor* function.

Algorithm 2 quadToolColor

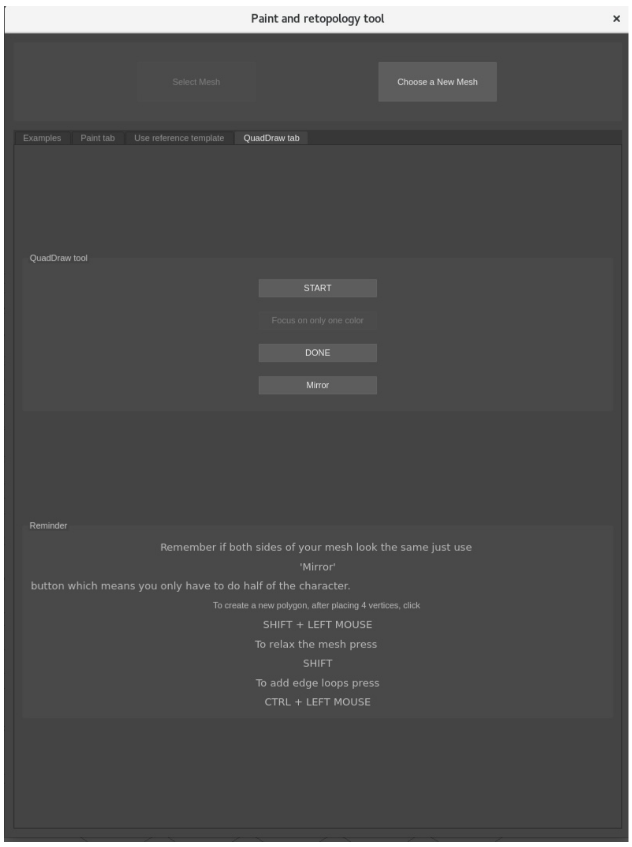
```
Check if there already exists layer called "color copy" and duplicate of the mesh called "colorCopy" with *deleteCopyColor*
    Select the duplicate
    Delete it
    Delete the "color copy" layer
    Set .copied to False
Make copy of the mesh, call it "colorCopy" and hide its layer
Select it
Group vertices into each color group with RGB value with *groupAndCopyVertices*
    Select duplicated mesh "colorCopy"
    Get the vertices of the mesh
    Make a group for a called color
    Get the color information of each vertex
    Group vertices into group depending on their color
    Select the group
    Convert selected vertices into faces
Delete selected faces
Select mesh "colorCopy" again
Paint all of its vertices into a color=(0.4, 0.4, 0.4)
Scale the "colorCopy" to (1.01, 1.01, 1.01) to make it bigger than the original mesh
Select the orignal mesh
Set .copied to True
```

When the *'Selects Mesh'* option is triggered all the vertices are being assigned the same color value that is not being used by any other colors (rgb= 0.4, 0.4, 0.4). This is done in order to avoid a situation when the unpainted area becomes part of the black-painted one since the initial color values of all vertices are rgb=(0.0, 0.0, 0.0) and color black has the same RGB.

The features described above are not available if the user chooses to use template since texture colors are not assigned to the vertices as the same values that this plug-in uses and therefore the *groupAndCopyVertices* would not work.

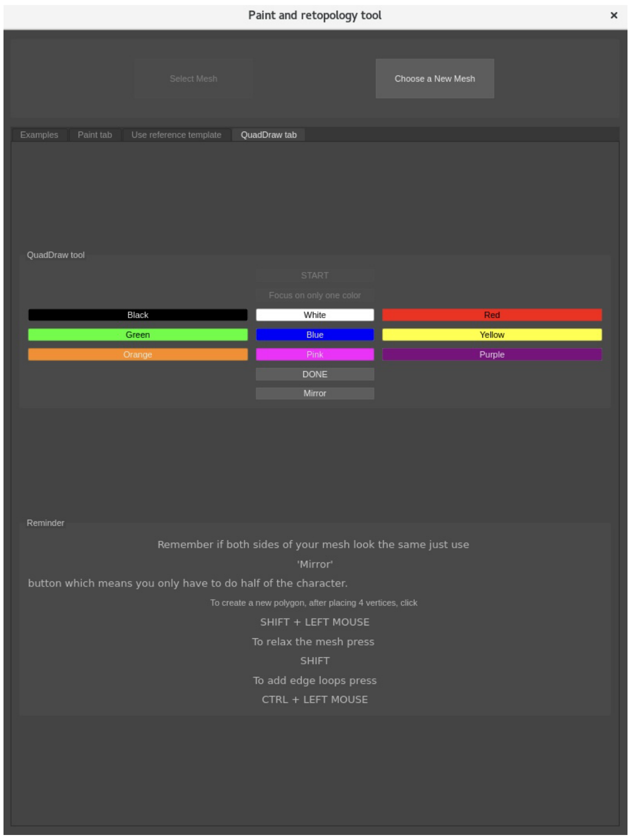
'DONE' button calls on functions that exit the QuadDraw tool, disables the Live mode and cleans up after all the other functions so there are no displayLayers or duplicated objects left but only the original mesh and new re-topolgised one. *'Mirror'* button should be used afterwords if the mesh was only done in half and was meant to be mirrored.

Reminder section is a simple QLabel with a text to remind the user how to use QuadDraw and what to press to create a new polygon.



Remember if both sides of your mesh look the same just use
'Mirror'
button which means you only have to do half of the character.
To create a new polygon, after placing 4 vertices, click
SHIFT + LEFT MOUSE
To relax the mesh press
SHIFT
To add edge loops press
CTRL + LEFT MOUSE

Figure 4.14: QuadDraw tab before enabling 'Focus on only one object'



Remember if both sides of your mesh look the same just use
'Mirror'
button which means you only have to do half of the character.
To create a new polygon, after placing 4 vertices, click
SHIFT + LEFT MOUSE
To relax the mesh press
SHIFT
To add edge loops press
CTRL + LEFT MOUSE

Figure 4.15: QuadDraw tab after enabling 'Focus on only one object'

There is also 'Choose a New Mesh' button that enables after 'Select Mesh' options is used and it resets the tool to the beginning so the user can start working on a new geometry.

4.4 Summary

In summary, this chapter presented an overview of the implemented design for the plug-in and its features created with variety of functions.

Chapter 5

Results

5.1 Results

To test if the initial goals have been achieved, two tests have been conducted on the same high-poly 3D model to compare the results. These can help proving the objectives of this project has been achieved. However, they cannot determine how much as there should be more tests done on different types of meshes because it can be expected that the higher the poly count the less effective the Paint tool option is as it has more calculations to be done. For the future, it would be useful to explore when the tool stops working depending on the density of the geometry.

Both tests were done with a head sculpted originally in Zbrush, 3D sculpting software, and re-meshed in it as well so the topology needs to be redone completely if the character were supposed to be used in production.

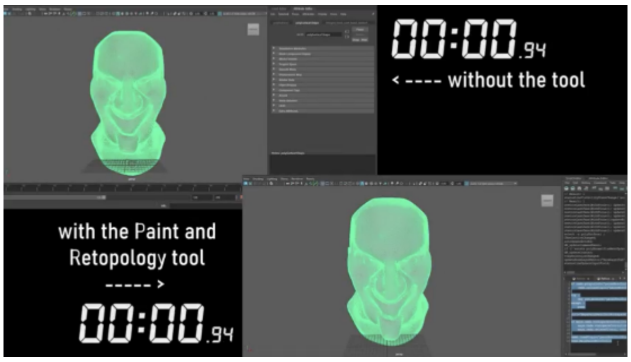


Figure 5.1: Both started with the same original high-poly mesh

First test was done with just the standard use of QuadDraw tool inside Maya and took around 15 minutes while the test involving the *Maya Paint and Retopology plug-in* took only around 12 minutes.

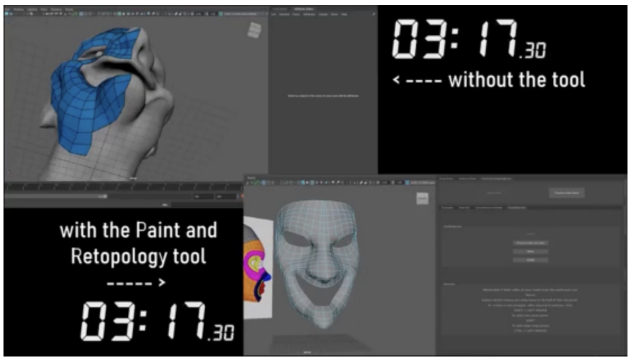


Figure 5.2: Final times sped up in comparison

It is also important to notice that in the initial test there were more mistakes done and there was more need for deleting newly created faces than in the other one where it was easier to follow the pre-painted areas and keep all the edge loops and the "extraordinary vertices" in the right places (Fig 5.3).

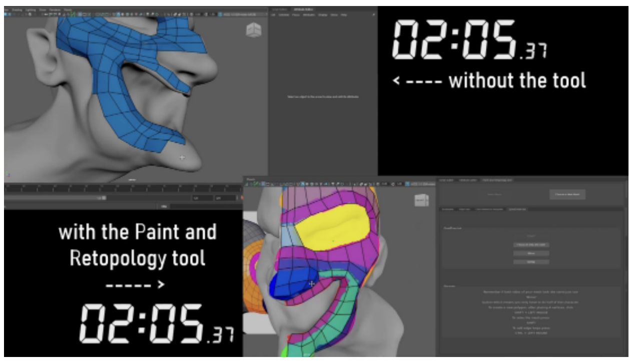


Figure 5.3: Different uses of valence points

On the other hand, the second test was done after the previous one which meant more familiarity with the mesh and knowing what mistakes to avoid from the previous try. Therefore this comparison is biased and therefore cannot be used as a definitive argument in favour of the plug-in's solution.

5.2 Conclusions

In conclusions, the conducted tests proved the final tool achieved at least one of its objectives but should not be used as conclusive argument supporting its usefulness since the results cannot be entirely objective. However, it is worth mentioning that the proposed method works as intended by not prolonging the pipeline, even hopefully cutting the time, and by not costing the artist many unwanted mistakes and has the potential to work even better as an educational tool.

Chapter 6

Conclusions

6.1 Summary

Topology is a vital part of the production pipeline because many other parts of it depends on it. Nowadays, to ensure the correct topology there are many tools available either for subdivision or re-topology but not all of them are fit for beginners. QuadDraw Tool from Autodesk Maya is one of them. It only offers the solution without much explanation of how the re-topology works. This is why this *Maya Paint and Retopology Tool* was created to help with this problem. Another reason was to also improve the pipeline in general with a 'painting on the mesh' workflow and reduce potential mistakes as much as possible and therefore not having a negative impact on the further processes in the pipeline, such as rigging and animating. The final output's result suggest that this has been achieved to at least some extent.

6.2 Evaluation

This project's main goals were to create a tool that can be easily navigated by novices and used as a proposed workflow for other artists to limit the number of mistakes and help with the amount of time spent on creating the right topology. This was achieved with a well-thought design of the user interface and functionalities in it. There are also explanations and reminders throughout the tool that serve as guidelines.

In conclusion, the final product has successfully met the initial criteria and has been developed with the user experience in mind. This can be used as an educational tool, as well as just another process for the artists to create new topology.

6.3 Future work

There are things that could be improved or explored further, such as allowing the user to input their own template in the 'Use reference template' tab. This could positively impact the interactivity of the tool and offer the artists more options to explore. Another idea that could further improve this tool is using ray casting as part of using the templates to project onto the mesh. However, with the short amount of time frame given for this project, this could not be achieved at the time because it requires a lot of time, tests and likely lower-level scripting in OpenMaya.

Overall, the final *Maya Paint and Retopology Tool* achieved the original aims and objectives. It also proved to be useful and has room for improvement.

References

1. DeRose, T., Kass, M., Truong, T., 1998. Subdivision Surfaces in Character Animation. Pixar Animation Studios. Available from: *https://graphics.pixar.com/library/Geri/paper.pdf* [Accessed: 10/08/2024]

2. Villemin, R., Hery, C., Konishi, S., Tejima, T., Yu, D., 2015.Art and Technology at Pixar, from Toy Story to today. Available from: *https://graphics.pixar.com/library/SigAsia2015/p* [Accessed: 10/08/2024]

3. Olsen, L., Samavati, F., Sousa, M., Jorge, J., 2009. Sketch-based modeling: A survey. Computers Graphics, 33(1), 85-103. Available from: *https://www.sciencedirect.com/science/artic* [Accessed: 10/08/2024]

4. Pixar, 2024. Modeling Tips. Available from: *https://graphics.pixar.com/opensubdiv/docs/mod$_n$ote* 10/08/2024]

Appendix

Appendix 1

Algorithm 3 paintVertices

```
import maya.OpenMaya as OpenMaya

import maya.OpenMayaMPx as OpenMayaMPx

import sys


nodeName = "paintVertices"

nodeId = OpenMaya.MTypeId(0x100fff)

class paintVertices(OpenMayaMPx.MPxDeformerNode):

"""

Commands —> MPxCommand

Custom —> MPxNode

Deformer —> MPxDeformerNode

"""


mObjpaintAttr = OpenMaya.MObject()

mObjweightAttr = OpenMaya.MObject()


def –init–(self):

OpenMayaMPx.MPxDeformerNode.–init–(self)

def deform(self, dataBlock, geoIterator, matrix, geoIndex):

input = OpenMayaMPx.cvar.MPxGeometryFilterinput

1. Attach a handle to input Array Attribute

dataHandleInputArray = dataBlock.inputArrayValue(input)
```

print(f"geoIdx: geoIndex")

2. Jump to particular element

dataHandleInputArray.jumpToElement(geoIndex)

3. Attach a handle to specific data block

dataHandleInputElement = dataHandleInputArray.inputValue()

4. Reach to the child - inputGeom

inputGeom = OpenMayaMPx.cvar.MPxGeometryFilterInputGeom

dataHandleInputGeom = dataHandleInputElement.child(inputGeom)

inMesh = dataHandleInputGeom.asMesh()

if inMesh.isNull():

raise RuntimeError("inMesh is None. Failed to get a valid mesh object.")

Envelope

envelope = OpenMayaMPx.cvar.MPxGeometryFilterEnvelope

dataHandleEnvelope = dataBlock.inputValue(envelope)

envelopeValue = dataHandleEnvelope.asFloat()

Paint

dataHandlePaint = dataBlock.inputValue(paintVertices.mObjPaintAttr)

paintValue = dataHandlePaint.asFloat()

Weight

dataHandleWeight = dataBlock.inputValue(paintVertices.mObjWeightAttr)

weightValue = dataHandleWeight.asFloat()

Colors and their weight values

```
colorWeightValues =
(0.0, 0.0, 0.0): 1.0,  black
(1.0, 1.0, 1.0): 2.0,  white
(1.0, 0.0, 0.0): 3.0,  red
(0.0, 1.0, 0.0): 4.0,  green
(0.0, 0.0, 1.0): 5.0,  blue
(1.0, 1.0, 0.0): 6.0,  yellow
(1.0, 0.5, 0.0): 7.0,  orange
(1.0, 0.0, 0.5): 8.0,  pink
(1.0, 0.0, 1.0): 9.0  purple



weight = colorWeightValues.get(paintValue, 0.0)
if weight is None:
raise RuntimeError(f"No value found for pValue: paintValue")

mFloatVectorArrayNormal = OpenMaya.MFloatVectorArray()
mFnMesh = OpenMaya.MFnMesh(inMesh)
mFnMesh.getVertexNormals(False, mFloatVectorArrayNormal, OpenMaya.MSpace.kObject)



mColorArray = OpenMaya.MColorArray()
mFnMesh = OpenMaya.MFnMesh(inMesh)
mFnMesh.getVertexColors(mColorArray)

print(f"Normal count: mFloatVectorArrayNormal.length(), Color count: mColorArray.length()")
```

```
while (not geoIterator.isDone()):
index = geoIterator.index()
weightValue = self.weightValue(dataBlock, geoIndex, index)
geoIterator.setPosition(geoIterator.position() * (1.0 - envelopeValue + weight * envelope-
Value))

if (mColorArray[geoIterator.index()] == paintValue):
mColorArray[geoIterator.index()] *= weight

geoIterator.next()

mFnMesh.setVertexColors(mColorArray)
print("all good")

Creator
def deformerCreator():
nodePtr = OpenMayaMPx.asMPxPtr(paintVertices())
return nodePtr

Initialize the script plug-in
def nodeInitializer():
"""
Create Attributes - check
Attach Attributes - check
Design Circuitry - check
```

```
  """
  mFnAttr = OpenMaya.MFnNumericAttribute()
  paintVertices.mObjPaintAttr = mFnAttr.create("PaintValue", "PVal", OpenMaya.MFnNumericData
  0.0)
  mFnAttr.setDefault(0.0, 0.0, 0.0)
  mFnAttr.setKeyable(1)
  mFnAttr.setMin(0.0)
  mFnAttr.setMax(1.0)

  paintVertices.mObjWeightAttr = mFnAttr.create("WeightValue", "WVale", OpenMaya.MFnNumeri
  0.0)
  mFnAttr.setKeyable(1)
  mFnAttr.setStorable(0)
  mFnAttr.setReadable(1)
  mFnAttr.setWritable(1)

  paintVertices.addAttribute(paintVertices.mObjPaintAttr)
  paintVertices.addAttribute(paintVertices.mObjWeightAttr)

  """
  SWIG - Simplified Wrapper Interface Generator
  """
  outputGeom = OpenMayaMPx.cvar.MPxGeometryFilter_outputGeompaintVertices.attributeAffect
paintVertices.attributeAffects(paintVertices.mObjWeightAttr, outputGeom)

Initialize the script plug-in
def initializePlugin(mobject):
```

```
mplugin = OpenMayaMPx.MFnPlugin(mobject, "Nina Kokot", "1.0")
try:
mplugin.registerNode(nodeName, nodeId, deformerCreator, nodeInitializer, OpenMayaMPx.MPxNode.

OpenMaya.MGlobal.executeCommand("makePaintable -attrType multiFloat -sm deformer
paintVertices paintVal")
except:
sys.stderr.write("Failed to register command: " + nodeName)

Unintialize the script plug-in
def uninitializePlugin(mobject):
mplugin = OpenMayaMPx.MFnPlugin(mobject)
try:
mplugin.deregisterNode(nodeName)
except:
sys.stderr.write("Failed to de-register command: " + nodeName)

input = OpenMayaMPx.cvar.MPxDeformerNodeInput
sel = OpenMaya.MSelectionList()
OpenMaya.MGlobal.getActiveSelectionList(sel)
dagPath = OpenMaya.MDagPath()
sel.getDagPath(0, dagPath)
```