CIS 510 Final Project

Areej Alghamdi and Jacob Lambert

Although we each implemented only certain features separately, we met several times to discuss the design and implementation of all the features. Jacob primarily worked on creating the face mesh, designing the Qt GUI, and setting up the callbacks in python. Areej primarily worked on creating individual deformers for each metric, and ensuring that the deformers worked together as required.

Maya Face Mesh

(Jacob) To create the face mesh, I first found a crude drawing of a human head to use as an image plane in Maya. The image plane helped with modeling realistic facial dimensions.

To begin the modeling process, I first created a polygon from the side perspective using the image plane. This polygon served as an outline of the head. After deleting extraneous faces, I extruded this outline to begin modeling.

For the eye and mouth areas, I created a cylinder and deleted unneeded surfaces. I then combined this cylinder with the outline mesh.

Throughout the modeling process, I attempted to use only quad faces. This required adding edge-loops and creating new polygons. Most of the polygons were created by extruding edges and merging vertices. I also used the `append-to-polygon` tool.

The eyes are spheres scaled to roughly match the size of the eye holes in the face mesh.

Self-critique

While the face does look somewhat human, it needs some improvement. Currently the model has no ears. Also, when smoothing the mesh, I did not distinguish between "soft" and "hard" edges, resulting in over-smoothing in some areas.

Maya Individual Deformations

(Areej) To achieve the desired chernoff faces I used combinations of wire, cluster, and blendshape deformers to deform duplicate instances of the base face.

I started with making duplicates of the base face and named them as follows: smileFace, frownFace, sadFace, BigFace, TallFace, WideEyes, CloseSetEyes, SpacedOutEyes, RaiseREyeBrow, RaiseLEyeBrow.

For the smileFace, and sadFace I used soft-select with small FallOff radius on Volume Falloff Mode to select the vertices around the mouth that I wanted to move in different directions to achieve the facial expression wanted (smile, or sad mouth expression). I made sure that the symmetry settings were ON (Object X) so the right side is exactly symmetrical to the left side of the face. I would select the vertices by hitting shift+click to add the control vertices of soft-selected ones, move them around, tweek my selection by deselecting, unitl I was satisfied with the final result. I would create a blendshape for the base face to this face and open the blendshape window to run the blendshpe and observe the deformation. If I saw that I needed to tweek the face, I would go back and improve the face.

For the frown face, I had symmetry on as well, soft-select OFF, and manually select every eyebrow vertex. I created a Cluster deformer and named

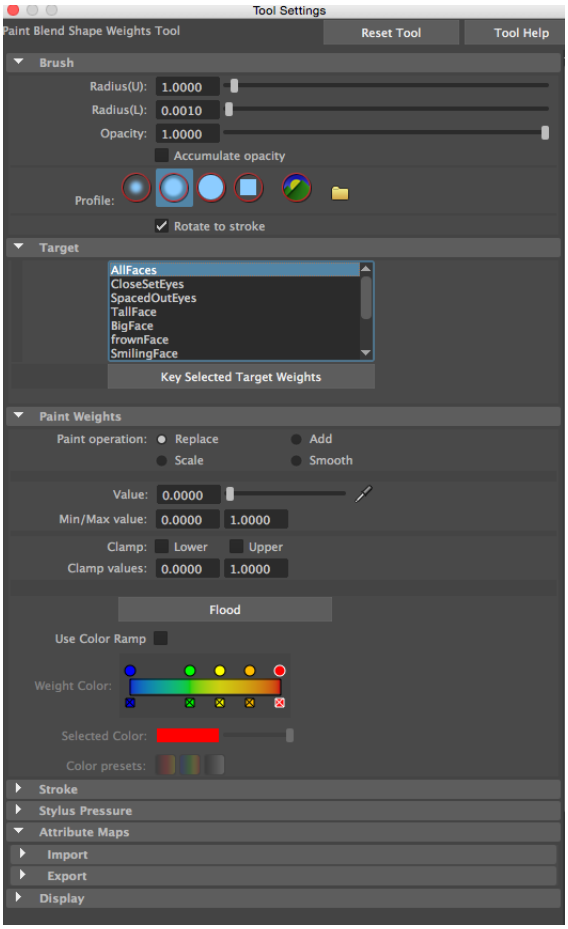
it frownEyebrow, I rotated, and moved the eyebrows closer to each other and around until I achieved the frown facial expression.

The same goes for the single raised Eyebrow faces, the only difference to the frown face steps is that the symmetry is Off.

For the bigFace I selected the whole face and then resized it by making it wider. I then created the blendshape by selecting the bigFace and then the base face and then go to the Deform window¿ click on BlendShape. A blendshape from the base face to the big face is created. To isolate the eyes, nose, and mouth from the rest of the face, I painted the weights of the blendshape. To prevent the eyes, nose, and mouth from resizing with the rest of the face, I had to paint the weight of the blendshape deformation on these areas to a value of Zero.

To Do That:
-go to Windows ¿ Animation Editors ¿ Blendshape.
-go to the blendshape you created from the base face to bigFace, move the controller to value 1(max value).
-Select the baseFace
-open the Deform window¿ go down to the Paint Weight section¿ click on the BlendShape select box.
- Select the target by clicking on bigFace, and select (Replace) from the paint weight section on the tool settings window and set the value to Zero, set the Brush size as desired.
- paint on the areas we dont want to be effected by the deformation, these areas will turn black.
- This painting work will make the face have uneven faces, so to make the face smoother, click on Smooth in the paint weight section of the tool Settings, choose a good brush size and paint around the the areas with value 0 to have smooth transitions between the surfaces of value 0 and value 1.
-So now to save these weights we painted on this face, go down to Attribute Maps, Click on Export.

The last step should have worked and I would have easily later just imported the weight map to any new blendshape but importing did not work. To save the weights painted I had to play around it because it is very time consuming to just do the repainting every time I create a new blendshape from the base face to extra faces. So in the blendshape control window, I set the controller of the bigface to the max value, duplicate the deformed face base, and move the duplicate, and rename it, bigFaceWeightPainted, now when I want to recreate the blendshape, I dont have to worry about losing the painted weights.

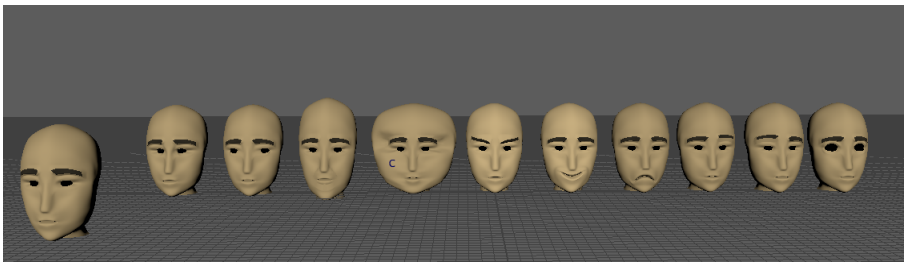


For the TallFace, I selected the whole face, turn Symmetry ON (Object X), Soft-selection OFF, and then manually deselect the vertices that I do not want to be resized. These vertices are the eyes, nose, eyebrows, and mouth. After that I resize the face by making it taller, and go back and tweek the selected vertices until no breakage is noticed. I created a blenshape and painted the weights around the areas I do not want to be affected. I followed by the smooth brush.

For the WidEyes, I had soft-select ON, Symmetry ON, and selected all the vertices around the eye opening, made a cluster, then re-sized it by making it wider.

For the CloseSetEyes and SpacedOutEyes, I had soft-select OFF, Symmetry ON, and manually select every vertex around the eye opening and move the vertices either to the right for the wideSet eyes effect or to the left for the close set eyes. I would move the selected vertices one bit at a time, add extra vertices that I have reached and move the how set of vertices again, I did that to prevent the breakage in the mesh from happening.

All the deformed faces and the base face are shown in the picture bellow.



Self-critique

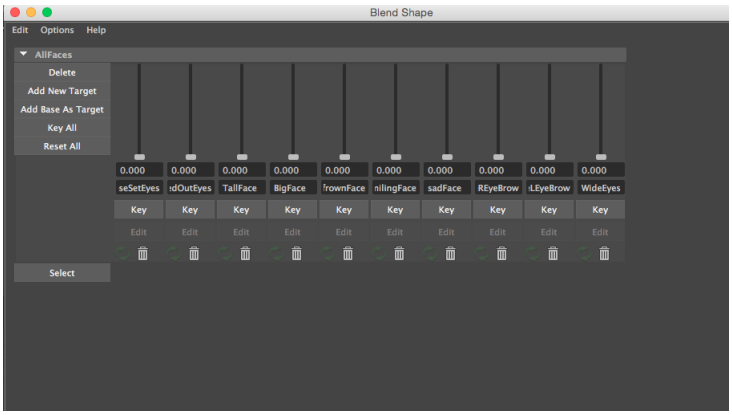
The steps I gave up, are the shortest and easiest way to achieve the same result of deformations. It was not as straight forward as this, it was rather a learning curve. The combinations of deformers and the order of applying them is not necessarily the best but they gave us the degree of deformations we seek in this project.

Maya Linking Deformations

(Areej)

As I mentioned earlier, I achieved the desired chernoff faces by using combinations of wire, cluster, and blendshape deformers to deform duplicate instances of the base face. For blenshapes to work well on this project, all faces deformed must be selected first, and the base face last and then the blendshape created. For the base face to blend smoothly between all the faces(each with different expressions), the blendshape must include all the faces. So for this project we have only one Blendshape.

To create the blendshape, select all the deformed faces, and select the base face the last, go to the Deform window, click on the blendshape select box, name the blendShape and click create. Now the blenshape is created, and can be manipulated through the GUI linked to it.

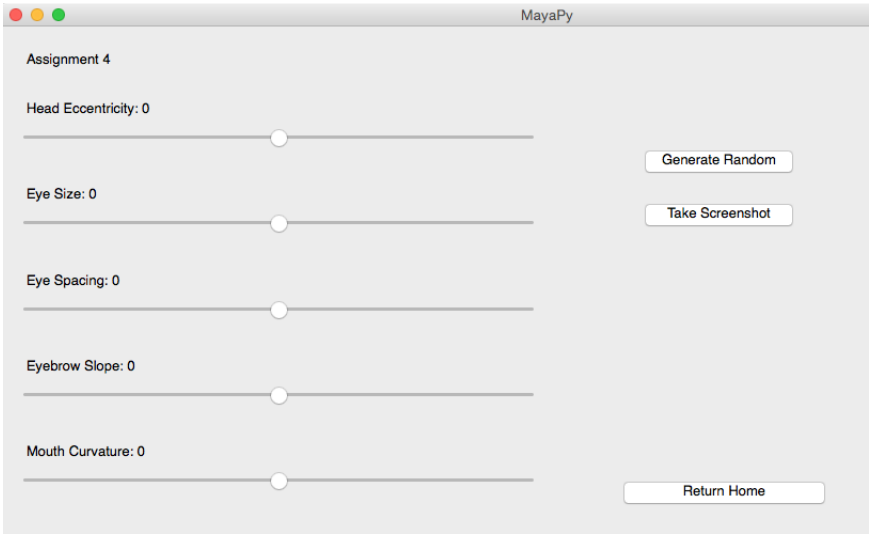


Self-critique

After sepnding a lot of time experimenting and researching best ways to combine multiple deformations on a single base face but insuring that unexpected behaviours are going to happen, we decided to use a blendshape deformer. Using a blendshape we were able to control the translations from one facial expression to another, and correctly combine multiple facial expressions.

Qt GUI

(Jacob) The Qt GUI consists is relatively simple. It consists of a text label and slider for each modifiable attribute. The label contains the name of the attribute and the current value assigned to the attribute. The slider ranges from -1 to 1, and is initialized to zero. The magnitude of the value of the slider represents the value of the associated attribute. The GUI also contains buttons to generate a random face and capture a screenshot.



Self-critique

The GUI could allow users to manually input values for the attributes in addition to using the slider. This would require a more MVC based design to keep the slider and user-input values consistent.

Python

(Jacob) Each element of the GUI has a corresponding event handler within in python. For the sliders, the handler gets the value from the slider, and applies the appropriate blend shape proportionally to the value. Some sliders have two associated blendshapes. For example, the `Mouth Curvature` slider applies the `Smile` blendshape for positive values and the `Frown` blendshape for negative values.

The sliders that affect the eyes also translate and scale the eye polygons, in addition to applying the appropriate blendshapes to the face mesh.

The `Generate Random` button generates random values for all of the sliders using python's random number generator. The `Take Screenshot` button allows you to select a region of the screen to save as `~/Desktop/faceN.png`, where `N` is the number of screenshots taken.

Self-critique

The translation of the eyes is currently dependant on the absolute position of the polygons in the world. It would be nice to somehow make this translation keyed or relative to the location of the face mesh. Additionaly the eye polygons do not match the mesh exactly for all face configurations.

Also, `Take Screenshot` is probably not portable, as it relies on the OSX specific `screencast` program.