BA Computer Visualisation and Animation

Innovations Into Computer Graphics

1

3D JellyFish development for Computer Graphics

Jack Kersey
-i7827320-

1 Sea Nettle Jellyfish, Source:http://28.media.tumblr.com/tumblr_lzfuh8Usd01qdxnvbo1_500.jpg

Table of Contents

1 **Abstract** – p3

2 **Introduction** – p3

3 **Jellyfish in Computer Graphics** – p4
- (i) Disney's Finding Nemo (Film) – Pink Pacific sea nettle (2003)
- (ii) Hotpoint Aqualtis Underwater World (TV) – Shirt Jellyfish (2008)
- (iii) Child of Eden (Game) – Abstract Jellyfish (2011)

4 **Research into the Structure of Jellyfish** – p7
- (i) The Pacific Sea Nettle
- (ii) The Make-up of 'Jelly'
- (iii) Movement of an Pacific Sea Nettle
- (iv) Tentacles and Oral Arms
- (v) Jellyfish Hood (Bell)

5 **Testing** – p11
- (i) Building a Test Rig
- (ii) Test Models used for Hood
- (iii) Animating the Hood with Two Deformers
- (iv) Tentacle Testing
  - (a) Method #1 – nCloth
  - (b) Method #2 – Paint Effects
  - (c) Method #3 – ikSpline

6 **Tutorial** – p24
- (i) Modelling and Texturing the Jellyfish Hood
- (ii) Modelling a simple large tentacle for nCloth or ikSpline.
- (iii) Rigging large tentacles with ikSpline Tool
- (iv) Rigging large tentacles with nCloth
- (v) Making Paint Effects for smaller tentacles
- (vi) Rigging and Animating the Body
- (vii) Putting Everything Together
- (viii) Potential Areas for Development

7 **Conclusion** –p50

8 **Bibliography and References** – p51

1) Abstract

The aim of this project is to produce a fully modelled, rigged and animated jellyfish which is convincing enough to be used in 3D Animation with a view to apply the techniques into my Major project. This paper contains research into existing Jellyfish in Computer Graphics and an in-depth study of real-life Jellyfish anatomy and behaviour. I have documented a series of tests used to determine the best way to make a 3D Jellyfish. The research findings are recorded in this paper, along with a written Tutorial showing exactly how I would create a jellyfish from modelling and texturing, to dynamics and animation.

2) Introduction

The aim of this project, for me, is to learn something new and develop something which hasn't necessarily been done before by any other students. I would like to develop a method for creating a Jellyfish in Maya that can be easily implemented by other students. This paper will concentrate on making life-like organic Jellyfish, and will strive to produce a method for doing so to the highest standard. The research on Jellyfish anatomy and behaviour I will learn from this paper will help me to re-produce the creature in 3D effectively. Through researching and understanding, I can establish what features of Jellyfish are elementary in defining them and incorporate them into my final tutorial and product. I will research existing Jellyfish in Computer Graphics Media, and also study images and videos of real-life Jellyfish. From this research I will run a series of Tests for making Jellyfish, from how to make the body and tentacles, to how to animate a jellyfish convincingly. Once the research is complete I will construct a tutorial explaining the methods I have discovered, and putting them into practice. From this tutorial, I want it to be clear and easy for any student to be able to make their own Jellyfish confidently.

3) Jellyfish in Computer Graphics

(i) Disney's Finding Nemo (Film) - Pink Pacific Sea Nettle (2003)

Probably the most memorable example of Jellyfish in Film is the Jellyfish produced in Disney's Finding Nemo. They managed to capture the delicate nature of Jellyfish, producing a very convincing and elegant result.

There were Several different Units working on specialised parts of the film, the Jellyfish scene was the job of the Ocean Unit. The Ocean Unit was responsible for such scenes as the school of moon fish, the angler fish chase, and the turtle drive.

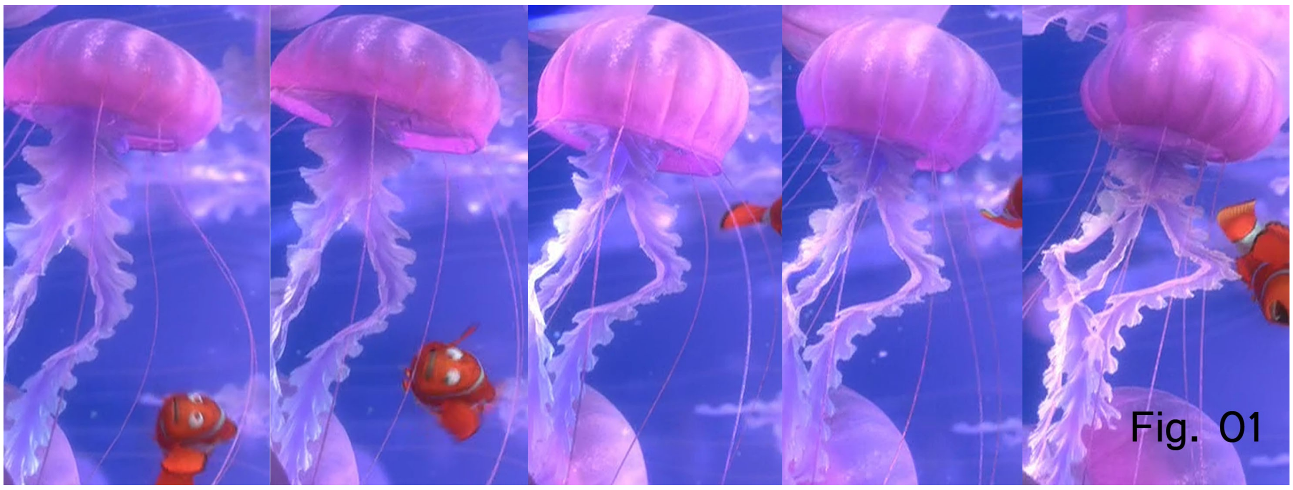


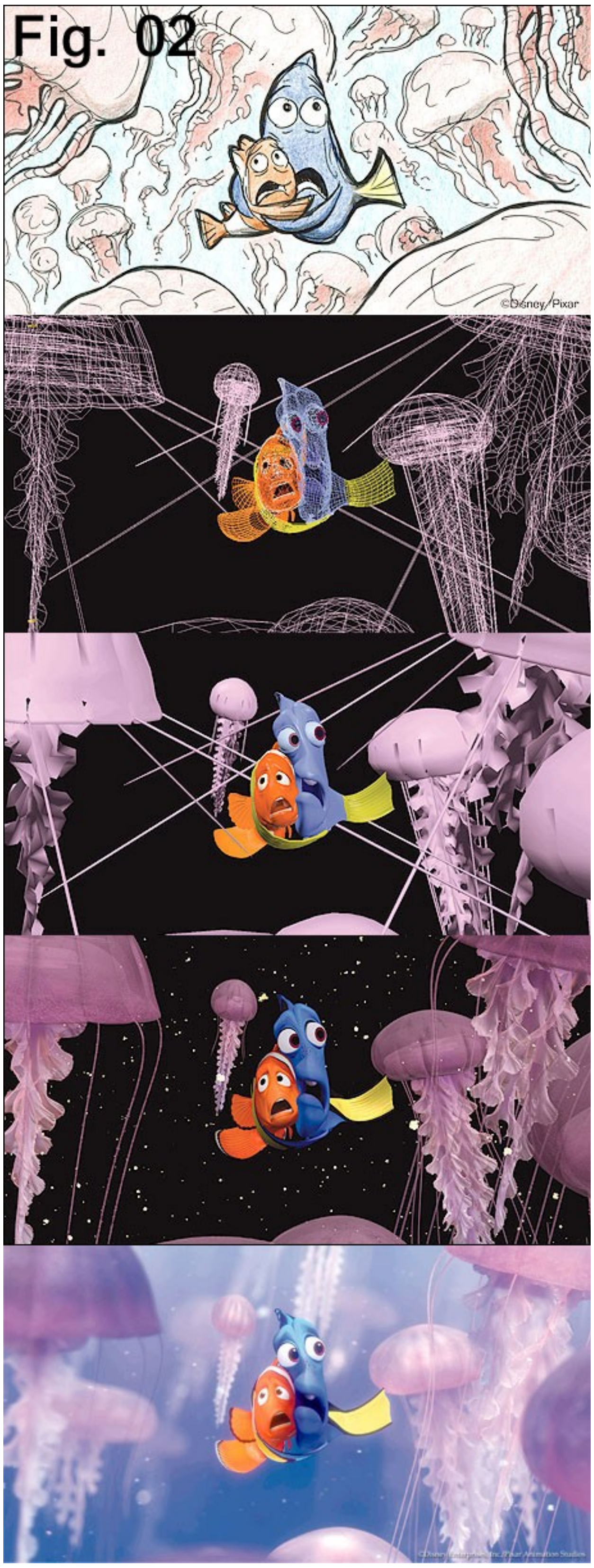
Fig. 01

The Ocean Unit admits that the most challenging scene was the Jellyfish Forest. As the Ocean Unit's CG supervisor, Lisa Forsell, explains, "This scene involved several thousand jellyfish. Our unit built the model for a single jellyfish and put a lot of work into the build-up of jellyfish density. This involved creating a simulation for the group that controlled the movement of the tendrils, how quickly they swam and in what direction. We had some great reference footage and were particularly fixated on one species from Palau that we found at the Monterey Aquarium. David Batte wrote a whole shading system we called 'transblurrency.' Transparency is like a window and you can see right through it. Translucency is like a plastic curtain that lets light through but you can't see through it. Transblurrency is like bathroom glass: You can see through it, but it's all distorted and blurry."[2]

2 CG Society *The making of Finding Nemo*

I would like to look at creating my own textures with a similar feeling to the one's seen in Pixar's Jellyfish forest. However, obviously I don't have the backing of several artists to produce such a high quality texture as they achieved, but I should be able to at least in principle produce my own version.

Fig. 02



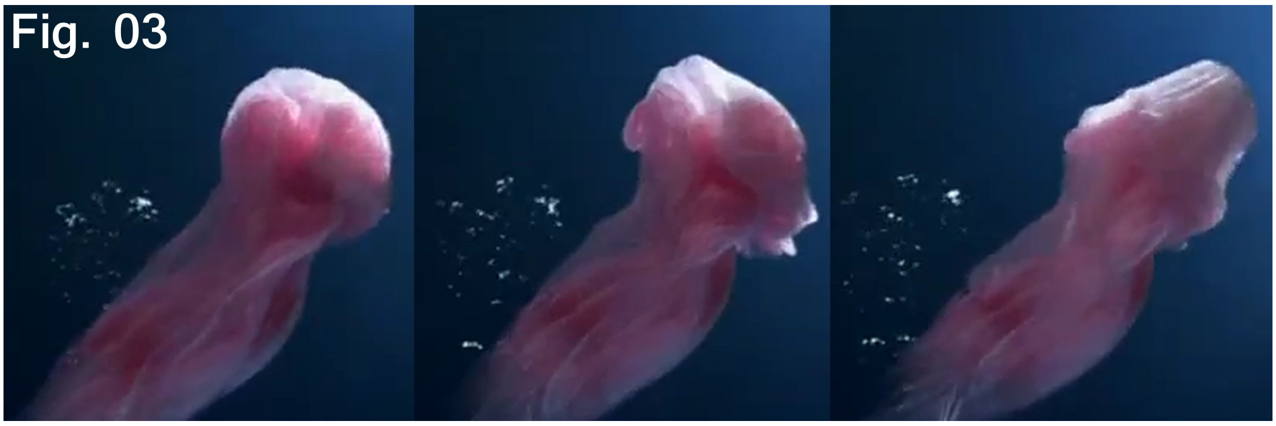
I took screen shots from the scene (Fig .01)[3] so I could see how Pixar had animated the jellyfish, especially the way the hood deforms. I found a series of Images from Finding Nemo (Fig. 02)[4] that show the stages of producing the scene. You can see that stage 4 is where the simulation is calculated, where before the tentacles were solid spikes, once simulated they become fluid and soft like real tentacles. Notice also that the Tentacles are not not part of the Hood geometry as you would expect, they are separate polygons, which are are constrained to the Hood at the Base.

3 Fig. 01 Walt Disney Picture's *Finding Nemo* (2003)

4 Fig. 02 CG Society Publication,*The making of Finding Nemo*

(ii) Hotpoint Aqualtis Underwater World (TV) - Shirt Jellyfish (2008)

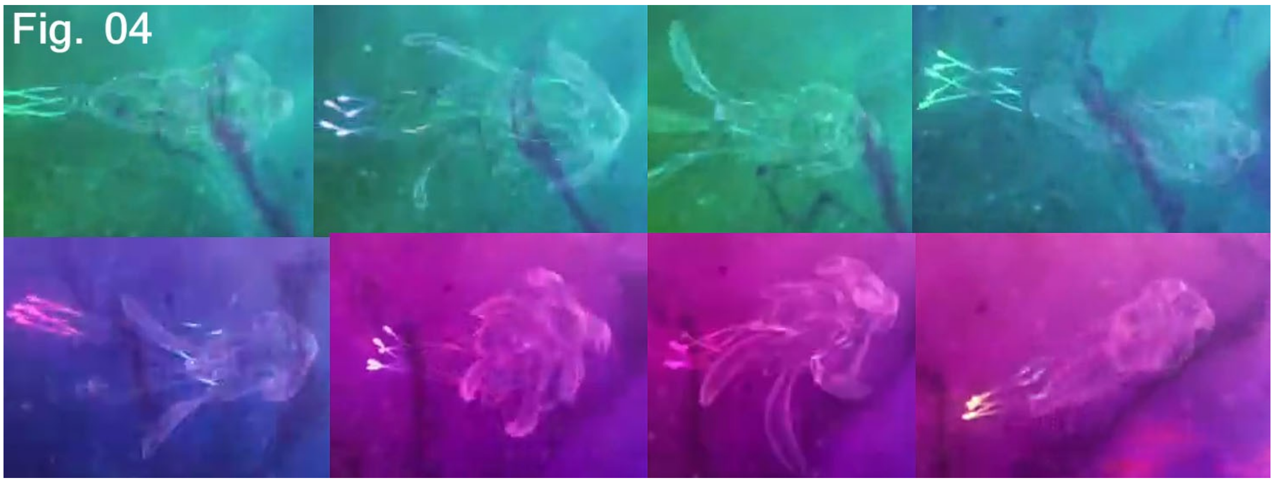
Fig. 03



The HotPoint advert is a perfect example of jellyfish movement being just as important as its anatomy and texture. The advert shows an 'underwater world' inside the washing machine, where the clothes come to life and replicate the movement of marine life. By replicating Jellyfish movement, they have been able to make this pink shirt look convincingly like a Jellyfish (Fig. 03)[5]. For me, this highlights the importance of getting the timing of my deformation and movement perfect, in order to convince the audience that they are looking at a Jellyfish.

(iii) Child of Eden (Game) - Abstract Jellyfish (2011)

Fig. 04



Similar to the Hotpoint Advert, the Child of Eden Jellyfish (Fig.04)[6] is a great example of Jellyfish movement. Although the Jellyfish is very abstract and tailored for a particular style, it has all the fundamentals of a real Jellyfish. The texturing is heavily influenced by transparency and glow effects, both of which are common traits in real Jellyfish. This approach to making Jellyfish isn't necessarily what I am trying to achieve but the simplicity of these Jellyfish models and textures inspired me to keep my Jellyfish simple.

5 Fig. 03 Source, Hotpoint , http://www.youtube.com/watch?v=2qdHwrYafuQ

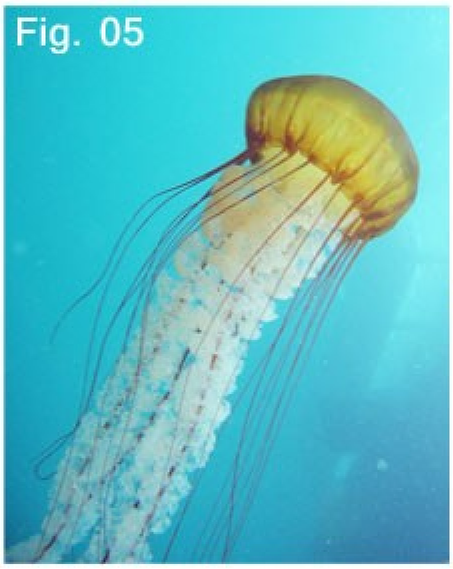
6 Fig. 04 Source, Child of Eden, http://www.youtube.com/watch?v=xuYWLYjOa_0&ob=av3e

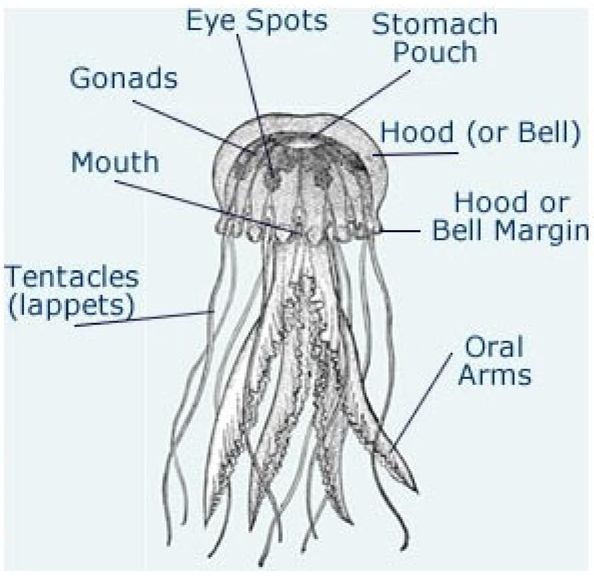
4) Research into the Structure of Jellyfish

There is no such thing as a typical Jellyfish because there is such a large variety in shape and size, from tiny species with diameters of less than one centimetre to ocean giants with diameters of up to 2 metres. For this project I'm going to be looking at one particular species as I could not possibly cover all varieties In one paper. The species I will be looking at in particular is the Pacific Sea Nettle or *Chrysaora fuscescens*[7].

(i) The Pacific Sea Nettle

The Sea Nettle (Fig. 05)[8] is equipped with long, thin tentacles for producing a deadly sting to it's prey. Its tentacles and frilly oral-arms are covered in stinging cells called *nematocysts*, which it trails behind itself catching small drifting animals in the open ocean. When the Jellyfish's tentacles latch onto its prey, they stick to the prey and paralyse it, the body is then pulled into the mouth by the oral arms where it is slowly digested. The bell, or hood, of the Sea Nettle can grow up to 45cm in diameter, with its oral arms and tentacles extending as far as 4.6 meters. For humans, its sting is often irritating producing a rash, but is rarely fatal. As the Sea nettle cannot chase after its prey it must eat as it drifts, relying on prey swimming into the path of its deadly tentacles.





The Jellyfish can be broken down into three categories for 3D production. The Hood (or bell), the Oral arms, and the tentacles. The combination of these three will make for a convincing 3D Jellyfish. The Eye spots, Mouth, stomach and Gonads can be imitated using textures.

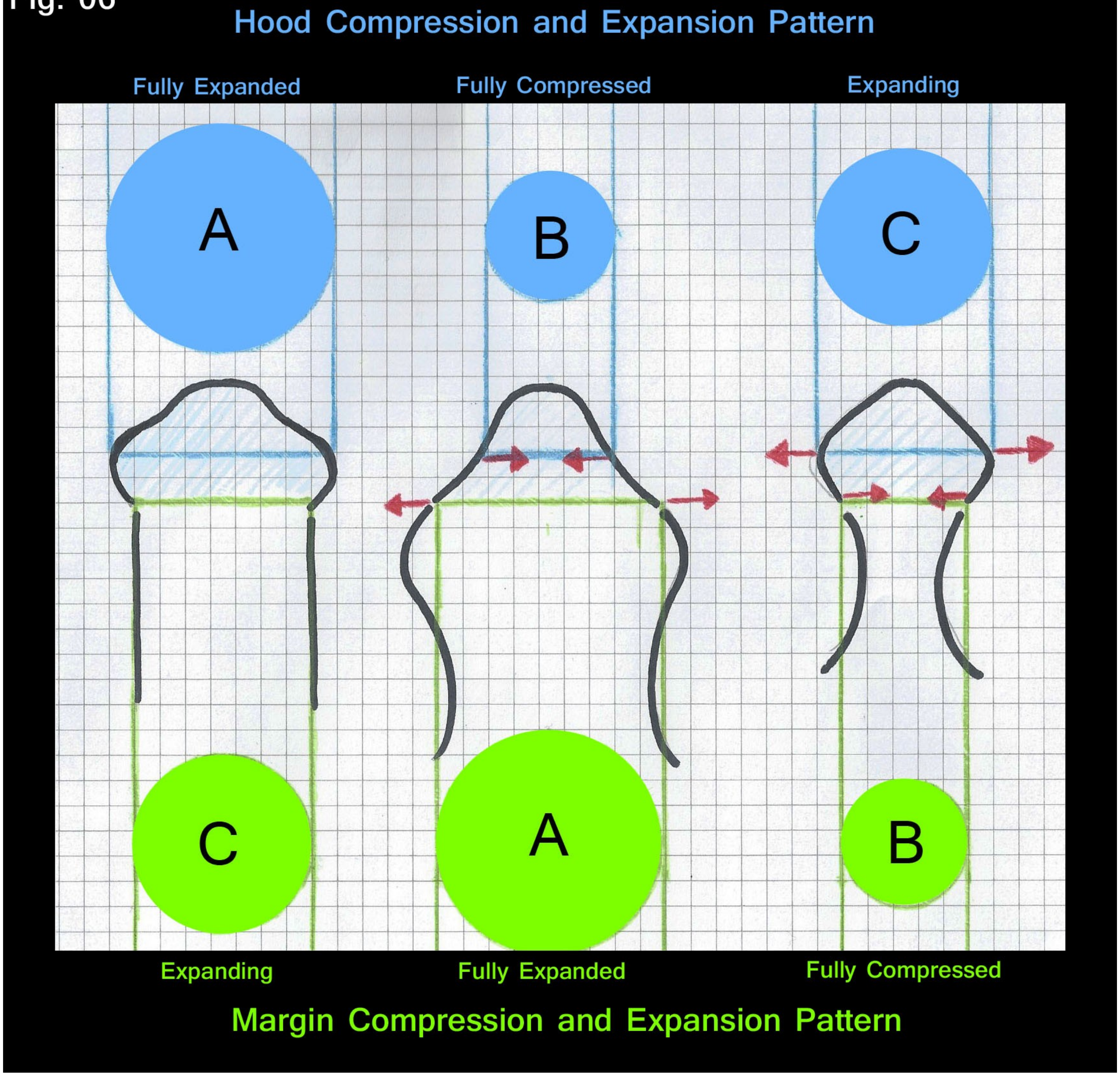
7. Monterey Bay Aquarium, Sea Nettle Exhibit
8. Jellyfish Anatomy http://www.animalcorner.co.uk/marine/jellyfish/jellyfish_anatomy.html

(ii) The Make-up of 'Jelly'

Jelly is a perfect material for the body of an animal that lives in water. It prevents jellyfish that live in deep waters from crushing and helps those that live near the surface to float. In order to achieve the 'Jelly' material for my Model I will use Maya Deformers to squash and stretch the model to imitate the soft Jelly membrane.

(iii) Movement of an Pacific Sea Nettle

Fig. 06

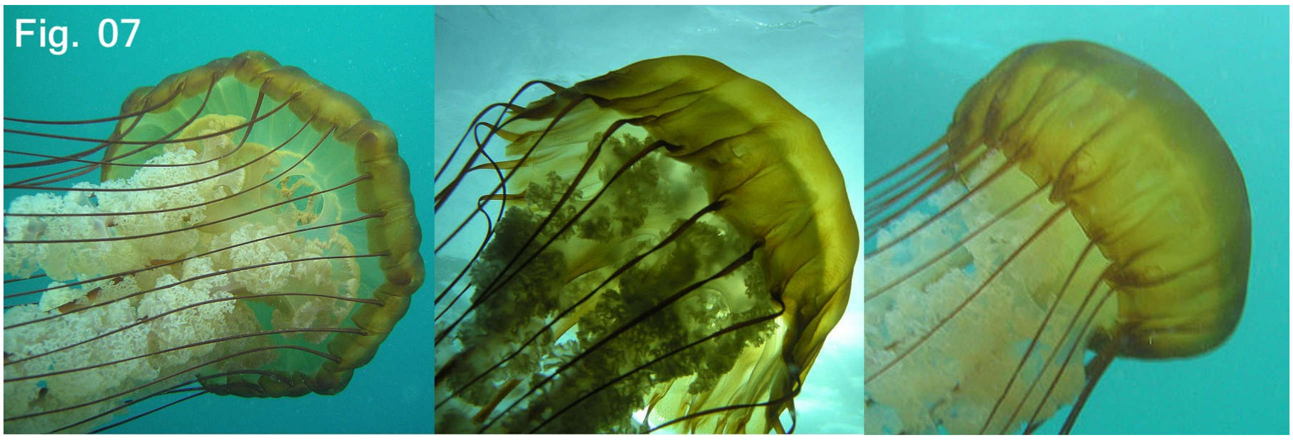


After I studied several videos and Images of Pacific Sea Nettles[9] I was able to construct a method for replicating their movement (Fig. 06). The secret to the way Jellyfish move is the way they drive water out from inside their Hood to create upward propulsion. The Jellyfish does this by expanding its Hood to its largest size creating a vacuum where water is forced in. Then it compresses it's Hood to force this water out, thus creating enough thrust to propel the Jellyfish through the water. From my Diagram above you can see that there are two main movements contributing to jellyfish movement. If you split the Jellyfish into two rings, one for its main Hood, and one for its Hood-Margin (Rim), where the tentacles are attached, its easier to understand the movement. As the Hood expands, the Margin is pulled out along with it, but with a delay. So as the Hood is expanding, the margin is still contracting. This offset movement creates the ripples in the tentacles, which follow the path of the Margin. As you can see from the diagram, whatever stage the Hood is at, the Margin will always follow but with a delay.

The pattern of movement for the Hood :	A-B-C-A-B-C	(Fig. 06)
The pattern of movement for the Hood Margin :	C-A-B-C-A-B	(Fig. 06)

The Margin is always one step back from the Hood (Fig.07)[10]. In order to create this sort of motion, I will use two squash deformers, one to control the Hood compression and the other to control the Margin compression. This way I can offset the animation of the two to achieve a realistic deformation.

Fig. 07



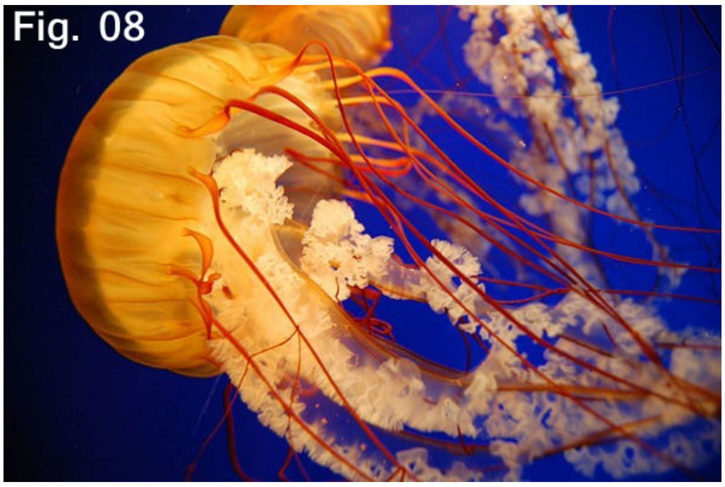
The linear movement of the Jellyfish occurs between the Hood Fully Compressed, and the Hood Expanding. This short period is when the Jellyfish moves in World space. As the Margin compresses the Jellyfish achieves propulsion and moves through the Water. When I come to Animating the Jellyfish I will have to animate the upward movement at the right time so that the effect of the vacuum is obvious and not just unnecessary movement.

9. Monterey Bay Aquarium, Source: http://www.youtube.com/watch?v=rzNI1Yh0F5A
10. Fig. 07 Wikipedia, Source: http://en.wikipedia.org/wiki/Chrysaora_fuscescens

(iv) Tentacles and Oral Arms

After studying several photos of Sea Nettle tentacles I thought of several ways in which they can be made in Maya. The thinner, darker tentacles (Fig. 08) attached to the Hood Margin will respond directly to movement generated by the latter. They must be flexible, and be able to simulate the ripple-effect caused by the Margin expanding and contracting. They could be done using Paint Effects, for which I won't need a model as Paint Effects can be converted to polygons after simulation. Another Option is for me to use nCloth or because of their snake-like deformation an ikSpline Joints chain. For these two methods to work I would need a simple polygon tentacle to create my simulations. The Frilly Oral Arms (see Fig. 08)[11] are less affected by the Hood and Hood Margin movement as they are attached to the mouth in the centre, these could be done using either nCloth or Paint Effects . In this paper I will test all three of these methods to find out what achieves the most realistic behaviour. A combination of the three methods is a possibility and may even be the only way to produce a realistic Jellyfish.

Fig. 08



(v) Jellyfish Hood (Bell)

Jellyfish are shaped much like an umbrella. The outside of which is called the *ectodermis,* the inside of which is called the *gastrodermis*. I will model the Hood using a revolved EPcurve, creating both the inside and the outside of the Hood at the same time. The colours of the Pacific Sea Nettle varies a huge amount, but they are usually shades of Ochre with some enjoying unique patterns on top of the Hood. I will use reference images to create the texture for my Nettle's Hood. Transparency is also Key to a Sea Nettle's anatomy, with some parts being semi-transparent and others being dense and non-transparent. I could use a transparency map to achieve this, or even look into sub-surface texturing. The movement of the Hood is also important, it should be obvious from the Hood movement that I have made a Jellyfish, even without tentacles.

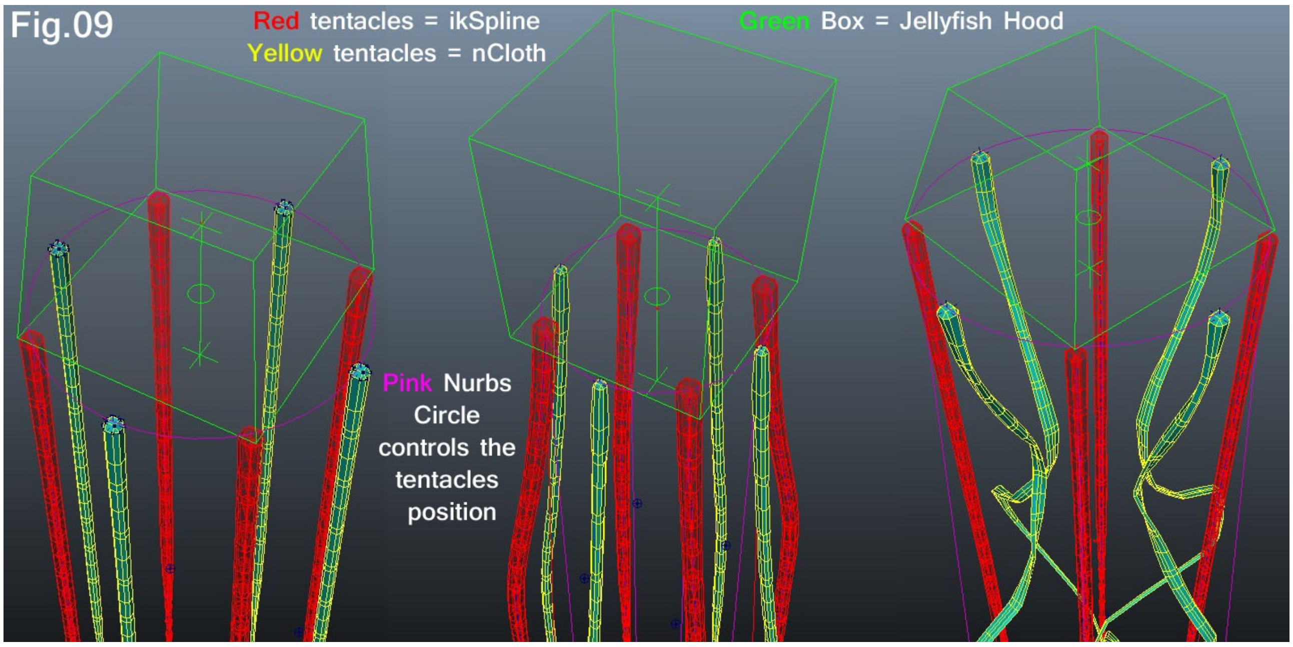
11 Fig. 08 Source : http://upload.wikimedia.org/wikipedia/commons/thumb/d/d6/Chrysaora_quinquecirrha.JPG/398px-Chrysaora_quinquecirrha.JPG

5) Testing

(i) Building a Test Rig

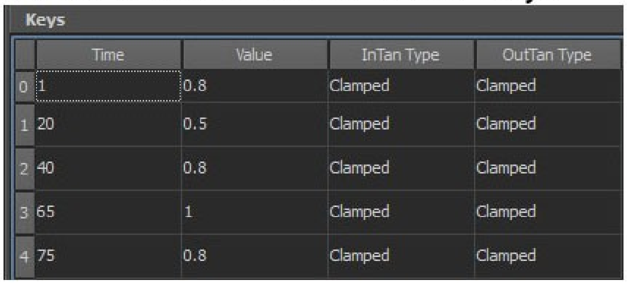
I need a Simple Rig (Fig. 09) to use as a yard stick for my tentacle tests. Jellyfish have a unique movement which can be replicated using simple Nurbs circles. By attaching any given tentacle to the edge of a Nurbs circle, and animating the circle to scale inward and outward like a Jellyfish head expanding and contracting, I can test the reaction of my tentacle rigs quickly and effectively.

Fig.09


Red tentacles = ikSpline
Yellow tentacles = nCloth
Green Box = Jellyfish Hood
Pink Nurbs Circle controls the tentacles position

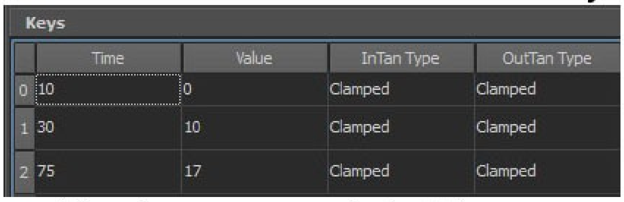
The expanding and contracting of a Jellyfish's Margin is what causes the tentacles to ripple. In order to replicate the contracting and expanding effect of the Margin, I used a Nurbs circle to represent my Margin. This would then be used to attach various types of tentacles for testing. By scaling the Nurbs circle, I was able to replicate the forces applied to real tentacles. The Nurbs circle was keyed to translate in the Y and keyed to scale in the X and Z axis (Fig. 10). I keyed the Nurbs circle's scale so that it replicated a real Jellyfish Hood's deformation. Now when I attach either an nCloth or an ikSpline tentacle it will follow the Nurbs circle up in the Y-axis and will also be pulled into the centre as the Nurbs circle scales down and vice versa when it scales up.

Nurbs circle XZ-Scale Value Keys


Keys

	Time	Value	InTan Type	OutTan Type
0	1	0.8	Clamped	Clamped
1	20	0.5	Clamped	Clamped
2	40	0.8	Clamped	Clamped
3	65	1	Clamped	Clamped
4	75	0.8	Clamped	Clamped

Nurbs circle Y-Translate Value Keys


Keys

	Time	Value	InTan Type	OutTan Type
0	10	0	Clamped	Clamped
1	30	10	Clamped	Clamped
2	75	17	Clamped	Clamped

All values were cycled at frame 75, using the Graph Editor

Fig.10

I mapped out where my Nurbs circle was moving in world space to demonstrate the behaviour I was producing. Fig 11 shows the Keyed path of the Nurbs circle, you can clearly see how its Y-translation and its XZ-scale values are creating a repeated movement pattern.

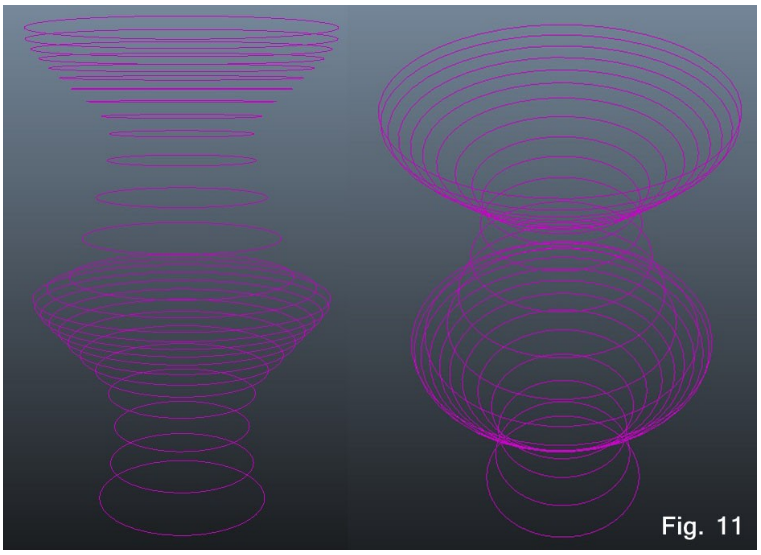


Fig. 11

(ii)Test Models used for the Hood

Fig. 12

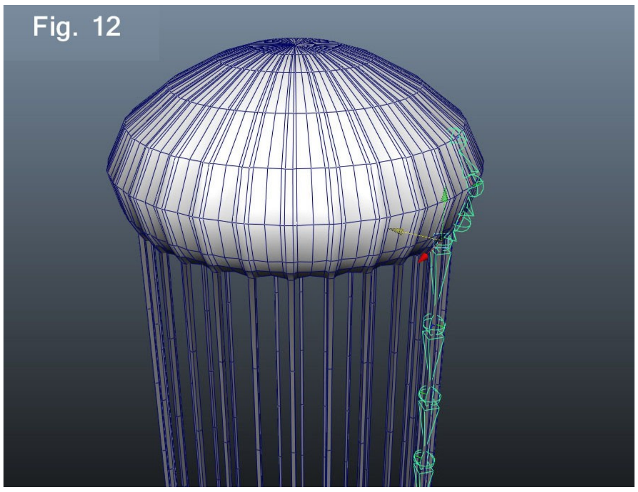
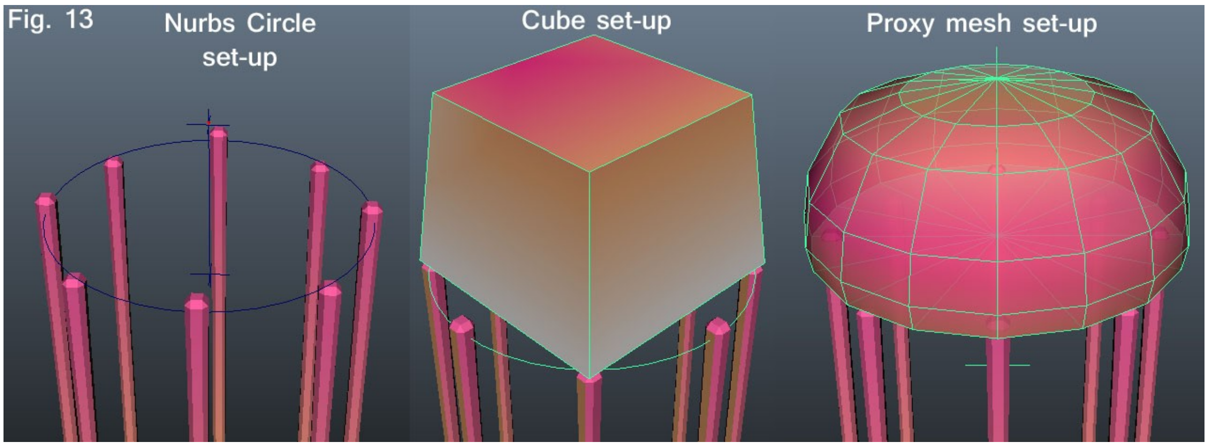
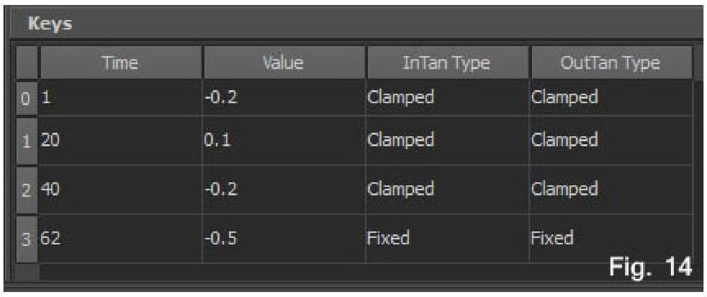

Before I decided to make the tentacles separate to the Hood I had experimented with rigging them as one polygon model.

Fig. 13


The idea was that I could attach a spline ik joint to every single tentacle running along the part of the Hood I wanted to deform. However I could not find a way to get my ik splines to react in the way I wanted using this method. I found that having the tentacles separate to the Hood gave me more control over the behaviour of them both. So using my Test Rig I attached a basic cube (Fig.13) to lay out the Y-axis animation, and the basic deformer animation. I could attach a new model at any time and it would follow the rig and the tentacles would react the same. This meant if I had any updates it was easy to incorporate them without re-animating everything. After I was happy with the movement of the Cube in the y-axis I replaced the cube with a proxy mesh of my Hood (Fig.13) . Once I had figured out the keying for the deformer handle (Fig. 14), I could replace the proxy model with the finished Hood once it had been modelled. I keyed the squash deformers *factor* only, its Y-axis movement was inherited from the proxy mesh it was parented to.

Test Rig - Squash deformer 'factor' key frames


Keys

	Time	Value	InTan Type	OutTan Type
0	1	-0.2	Clamped	Clamped
1	20	0.1	Clamped	Clamped
2	40	-0.2	Clamped	Clamped
3	62	-0.5	Fixed	Fixed

Fig. 14

The tentacles were attached to the nurbs circle differently depending on the type created. Tentacles made using ikSpline had to be attached by parenting the original spline Nurb curve to the nurbs circle. Tentacles made using nCloth had to be attached by suing a translate constraint on the top few vertices. Tentacles made using Paint effects could either be painted directly onto the proxy mesh, or alternatively a transparent plane could be used as a canvas, which was then parented to the Nurbs circle.

(iii) Animating the Hood with Two Deformers

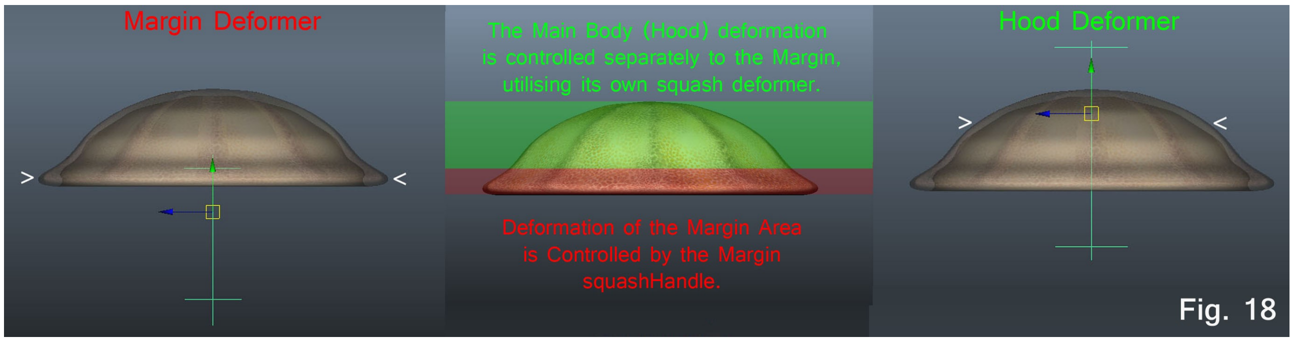

Margin Deformer
The Main Body (Hood) deformation is controlled separately to the Margin, utilising its own squash deformer.
Deformation of the Margin Area is Controlled by the Margin squashHandle.
Hood Deformer

Fig. 18

As I discovered in my research, there is a delay between the Hood compressing and the Margin Compressing. In order to achieve this effect, I used two squash deformers (Fig.18), one affecting the Margin (the bottom edge), and the other affecting the Main body or Hood.

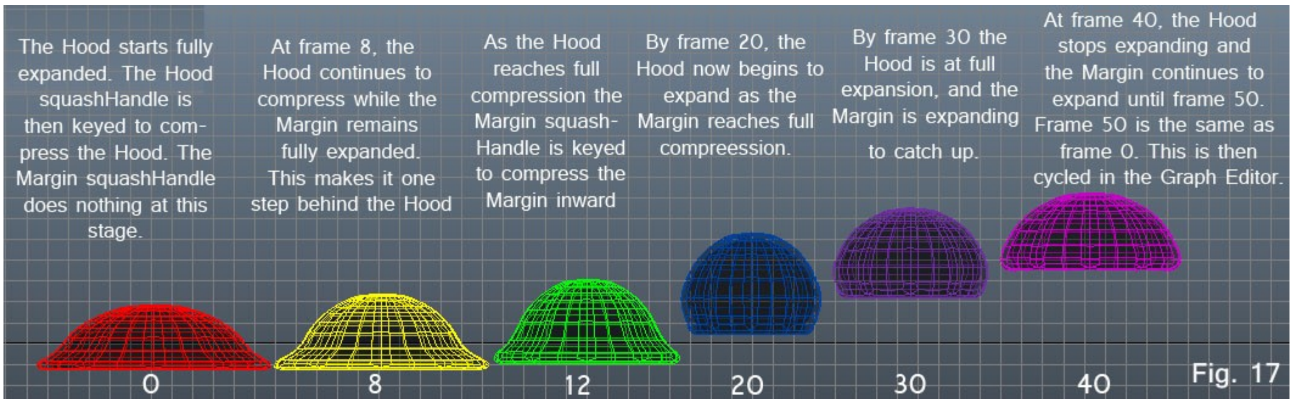
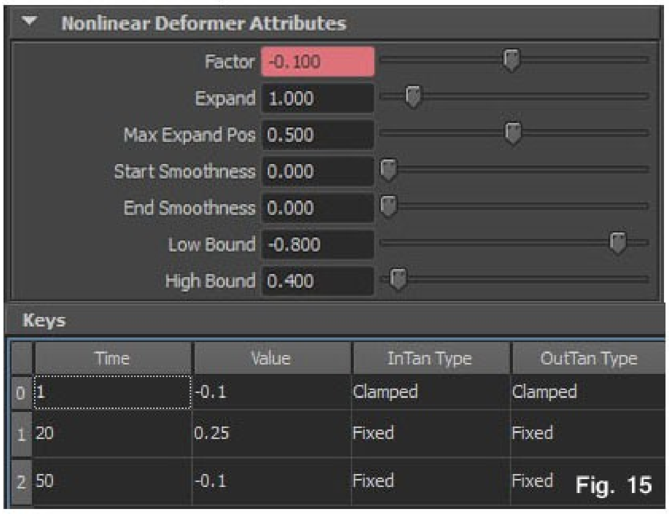

The Hood starts fully expanded. The Hood squashHandle is then keyed to compress the Hood. The Margin squashHandle does nothing at this stage.
At frame 8, the Hood continues to compress while the Margin remains fully expanded. This makes it one step behind the Hood
As the Hood reaches full compression the Margin squash-Handle is keyed to compress the Margin inward
By frame 20, the Hood now begins to expand as the Margin reaches full compreession.
By frame 30 the Hood is at full expansion, and the Margin is expanding to catch up.
At frame 40, the Hood stops expanding and the Margin continues to expand until frame 50. Frame 50 is the same as frame 0. This is then cycled in the Graph Editor.
0
8
12
20
30
40

Fig. 17

The Hood has to squash a few frames before the Margin (see Fig. 17), this way the Margin is always playing catch-up. Negative numbers will expand the squash deformer, while positive numbers will contract it. If you look at the keyed settings for both deformers, the Margin squash handle is expanding between frame 0 and 10, while the Hood squash handle is doing the opposite and contracting continuously from 0 to 20. The Margin handle only starts contracting between 10 and 20 where the value jumps from -0.4 to 0.2. This ten frame advantage the Hood deformer has over the Margin deformer, simulates the behaviour of real Jellyfish. Between frame 20 and 50 both deformers slowly expand back to their starting position, with the Margin expanding 10 frames behind the Hood.

This is the settings I used for my final Model (Fig 15-16) and intend to use in my tutorial as they were so effective. The test animation was cycled using the Graph editor at frame 50 where the values equalled those at frame 1.

Hood squashHandle Attributes



Nonlinear Deformer Attributes

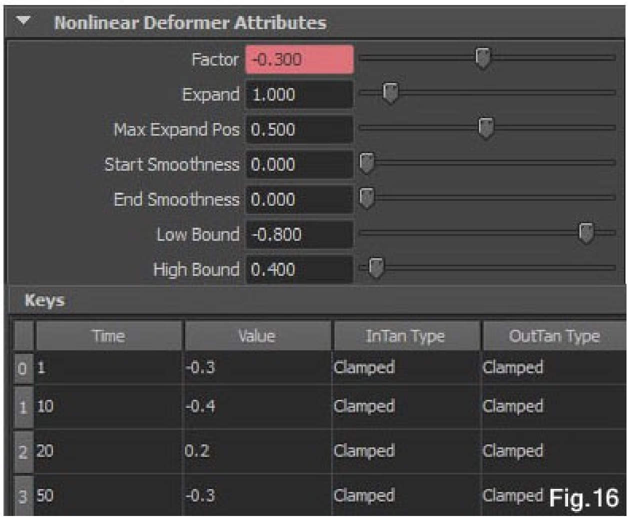
Factor	-0.100
Expand	1.000
Max Expand Pos	0.500
Start Smoothness	0.000
End Smoothness	0.000
Low Bound	-0.800
High Bound	0.400

Keys

	Time	Value	InTan Type	OutTan Type
0	1	-0.1	Clamped	Clamped
1	20	0.25	Fixed	Fixed
2	50	-0.1	Fixed	Fixed

Fig. 15

Margin squashHandle Attributes



Nonlinear Deformer Attributes

Factor	-0.300
Expand	1.000
Max Expand Pos	0.500
Start Smoothness	0.000
End Smoothness	0.000
Low Bound	-0.800
High Bound	0.400

Keys

	Time	Value	InTan Type	OutTan Type
0	1	-0.3	Clamped	Clamped
1	10	-0.4	Clamped	Clamped
2	20	0.2	Clamped	Clamped
3	50	-0.3	Clamped	Clamped

Fig.16

(iv) Tentacle Testing

(a) Method #1 Using nCloth

What Is nCloth?

nCloth is the first implementation of Maya Nucleus, Autodesk's simulation framework. nCloth gives the artist further control of cloth and material simulations. When the nCloth object is created, an nClothShape and a Nucleus node are created. It can be used for obvious things such as clothes and flags, but can also be used effectively when simulating other dynamic movement such as building destruction and explosions, flower petals and leaves, or even putting dents in objects. I decided to look into using nCloth because I wanted a soft cloth-like material to be created for my tentacles, and therefore nCloth could be a useful tool.

Method - for building single tentacle[12]

-First I created a proxy mesh of my tentacle for testing
-Next I created the nCloth shape with nMesh->create nCloth
-We now have an nClothShape and a Nucleus node to modify and affect.
-I then shift selected the vertices at the top of my mesh, and applied a *Transform constraint.*[13] The Dynamic Constraint can now be parented(Ctrl-P) to my Test Rig for a test Simulation.

What I found when I attached it to my Rig

I attached several nCloth tentacles to my Test Rig to see what was happening. The movement of the nCloth took some time to cache, but once it had finished it had a pretty convincing fluid movement. I used a couple of turbulence fields to encourage the tentacles to move more independently of each other. Without the turbulence field they all had identical simulations (Fig. 19), which individually looked good, but collectively they were far too symmetrical and thus unconvincing.

12. **E. Keller** *Mastering Maya 2011,* p296
13. **E. Keller** *Mastering Maya 2011,* p723-725

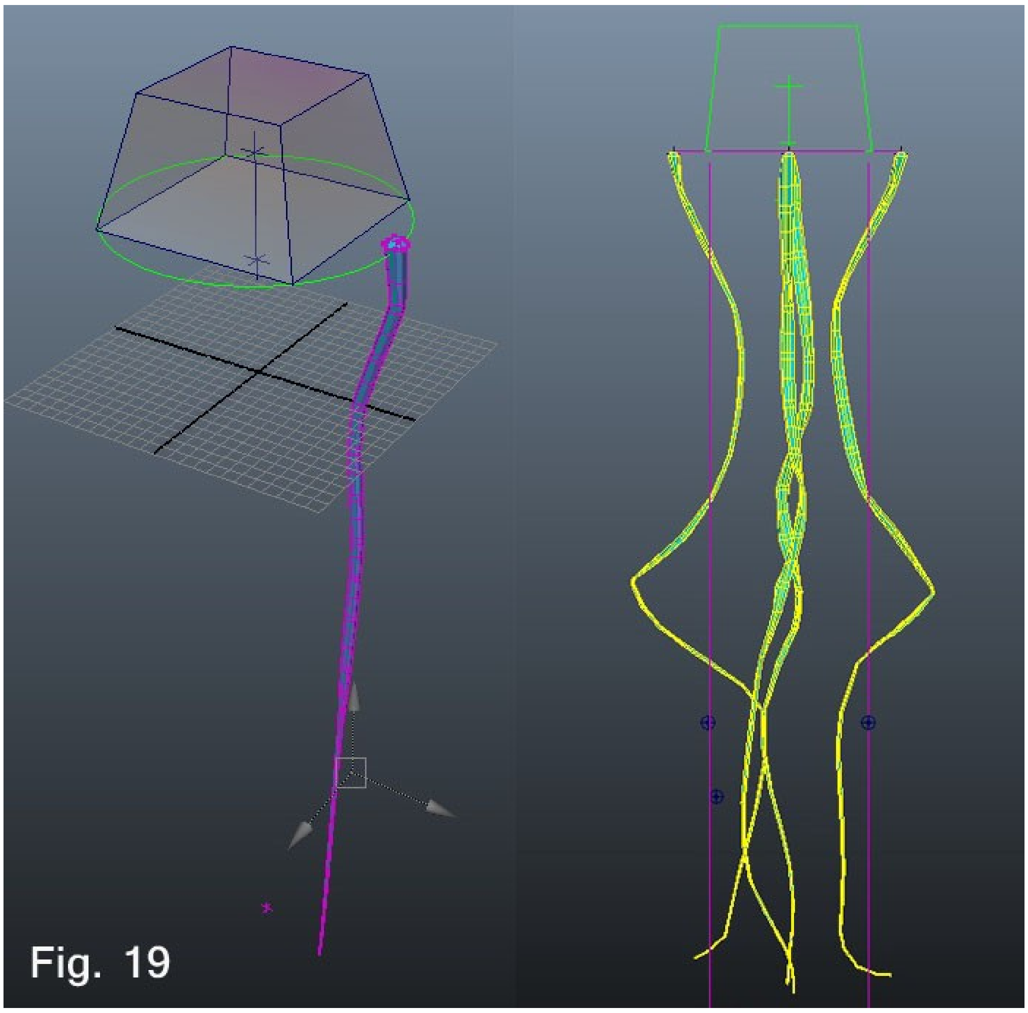


Fig. 19

nCloth Problems

One problem I discovered was a large degree of delay on the nCloth's simulation when the rig is animated (Fig. 20). This is caused by the nucleus solver not being able to keep up with the Transform constraint's movement. To solve this we must cache the nCloth simulation, this way the solver has already calculated the cloth movement, and so the mesh will keep up with the constraint without any delay.

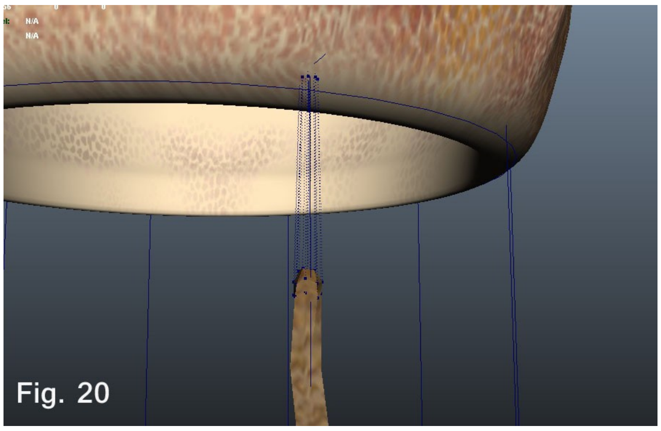


Fig. 20

(b) Method #2 Using Paint Effects[14]

-attached to body of Jellyfish

What Is Paint Effects?
The Paint effects tool is much like a real paintbrush, in that, when you drag it you create a stroke on screen. A stroke is a collection of curves with attributes that define how the paint is applied along the stroke path. It can be used to paint on a 2D canvas and create 2D textures, or it can also be applied in 3D.You can create environments, paint trees, flowers, grass, clouds and various other presets. I want to use this to create some simple painted 3D tentacles for my jellyfish, which will be attached to a polygon surface (the jellyfish body).

Method – for building a collective of tentacles
I tried out a few different settings to see what I could produce using paint effects. In order to create tentacles I set the Template Brush to draw 'tubes'. Tubes are a very useful tool within Paint Effects, and come with a large amount of attributes which means there is a lot of control over their behaviour and creation.

- First I Created a polygon Plane to act as my Paintable Jellyfish body.
- Then In the Paint Effects menu I reset the Template Brush settings to default.
- I wanted to experiment with making a few different tentacles, I tried out a variety of settings to produce long, thin tentacles and short and wide Oral arms.
- In order to give the tentacles the impression of being under water I reduced the gravity to 0.1 (*Behaviour->Forces->Gravity)* and the Momentum to 0. Also under Behaviour I changed Length Flex to 0.4.
- I wanted the tentacles to wriggle about slowly, so I used a turbulence field, set to World Displacement->Smooth over Time and Space. I used the Following turbulence field settings on all my Paint Effects

Paint Effects Turbulence Settings

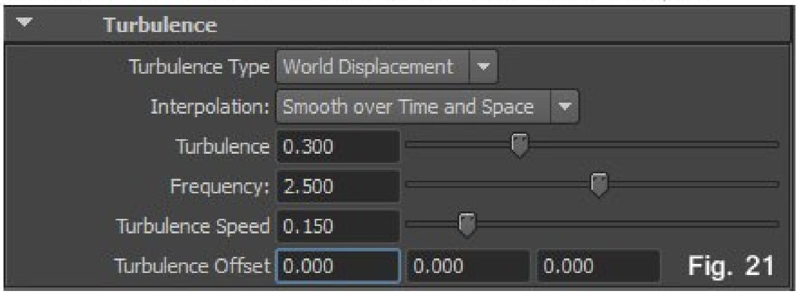
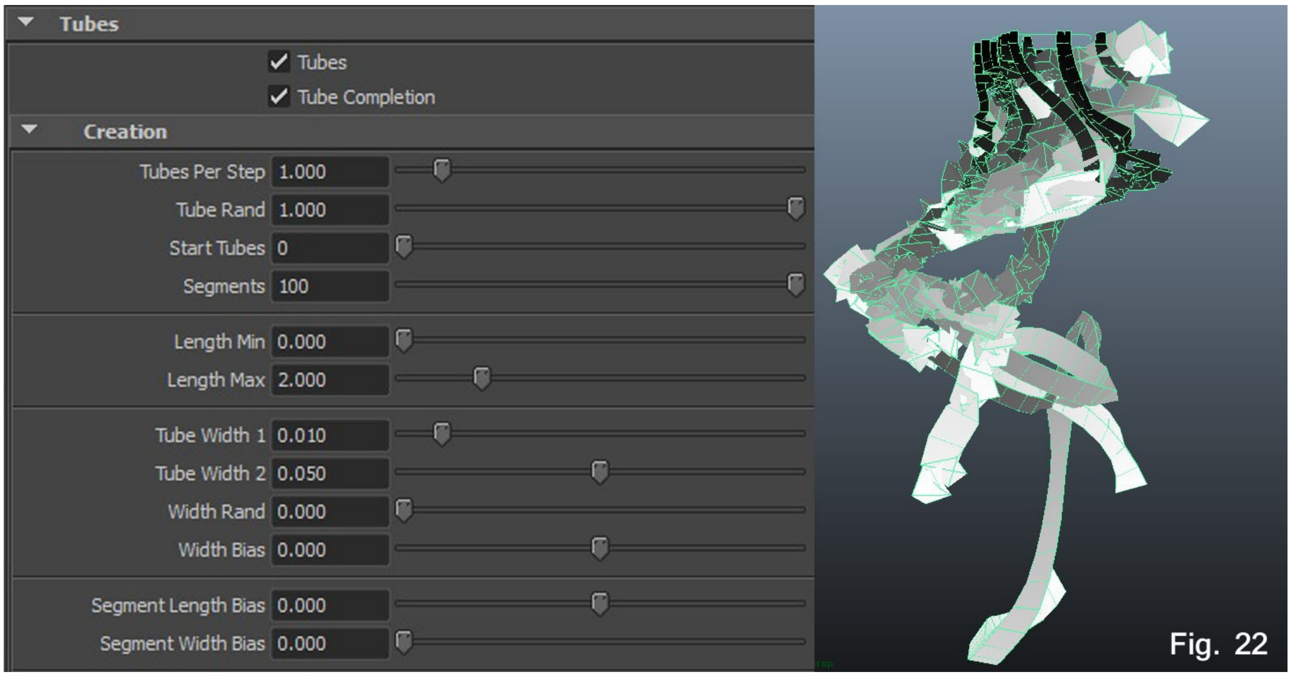

Turbulence
Turbulence Type: World Displacement
Interpolation: Smooth over Time and Space
Turbulence: 0.300
Frequency: 2.500
Turbulence Speed: 0.150
Turbulence Offset: 0.000 0.000 0.000

Fig. 21

The low turbulence speed gives the impression the tubes are floating gently under water. The higher the turbulence, the faster Paint Effects simulate movement. I made Four different types of Paint Effects tentacles, all of which I could use on my Jellyfish.

14. **Alias WaveFront**, *Using Maya : Paint Effects,* p22, p110-176

Type #1 – Creation Settings (Fig. 22)Analysis



Tubes

✓ Tubes

✓ Tube Completion

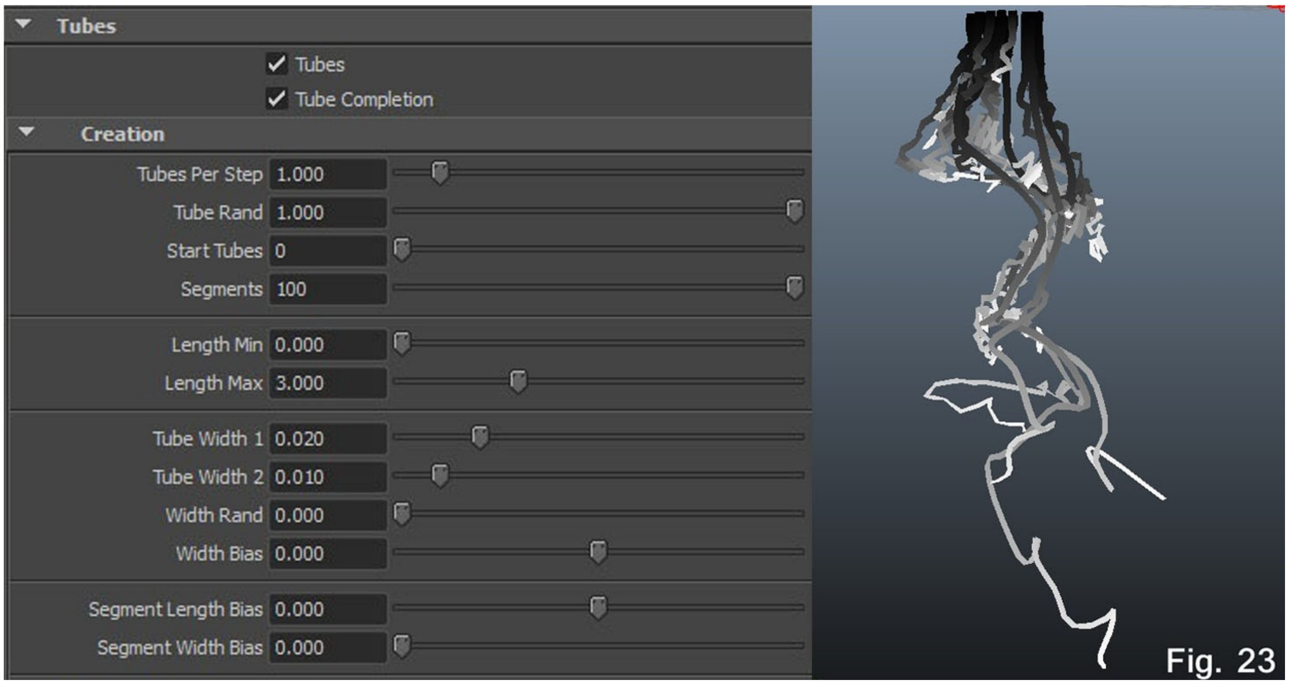
Creation

Tubes Per Step	1.000
Tube Rand	1.000
Start Tubes	0
Segments	100
Length Min	0.000
Length Max	2.000
Tube Width 1	0.010
Tube Width 2	0.050
Width Rand	0.000
Width Bias	0.000
Segment Length Bias	0.000
Segment Width Bias	0.000

Fig. 22

By making the starting width smaller than the end width and using a high tube per step value I was able to create a very dense and thick collection of tentacles. These types of tentacles would work well as Oral arms because of their size and shape and they behave more like a think cluster than separate strands of tentacles.

Type #2 – Creation Settings (Fig. 23)



Tubes

✓ Tubes

✓ Tube Completion

Creation

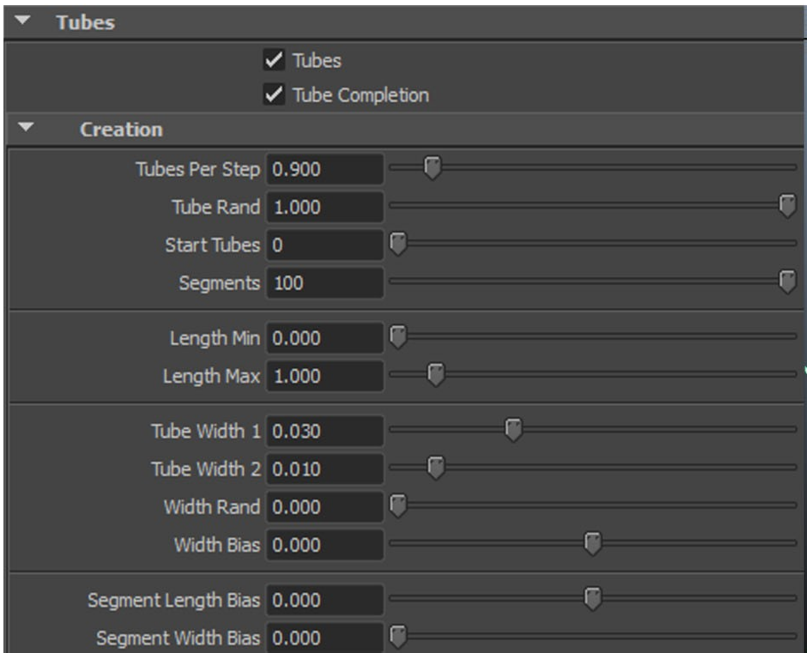
Tubes Per Step	1.000
Tube Rand	1.000
Start Tubes	0
Segments	100
Length Min	0.000
Length Max	3.000
Tube Width 1	0.020
Tube Width 2	0.010
Width Rand	0.000
Width Bias	0.000
Segment Length Bias	0.000
Segment Width Bias	0.000

Fig. 23

Analysis

This type was simply a longer thinner variation on Type #1. I liked the look of this tentacles because its a nice compromise between the heavy Type #1 and the light Type#4.

Type #3 – Creation Settings (Fig.24)


Tubes
✓ Tubes
✓ Tube Completion
Creation

Tubes Per Step	0.900
Tube Rand	1.000
Start Tubes	0
Segments	100
Length Min	0.000
Length Max	1.000
Tube Width 1	0.030
Tube Width 2	0.010
Width Rand	0.000
Width Bias	0.000
Segment Length Bias	0.000
Segment Width Bias	0.000

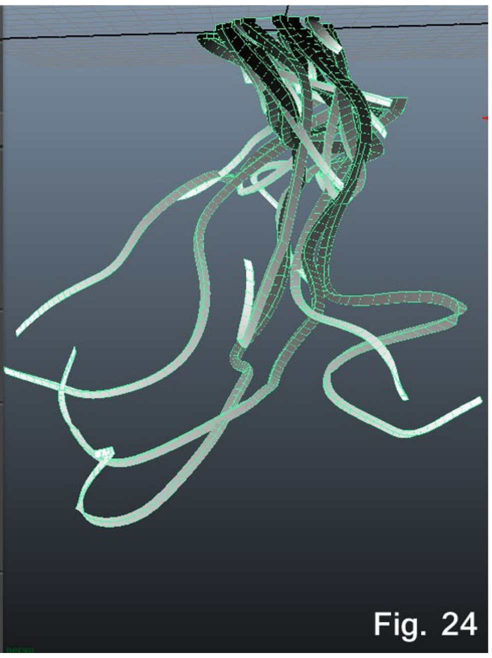
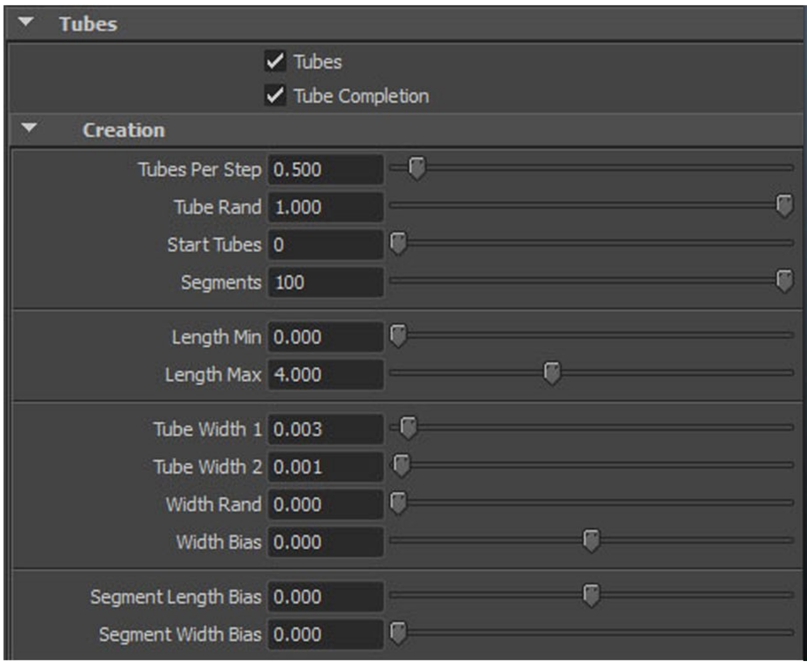


Fig. 24

Analysis
This type of tentacle behaved and looked more like thick limbs than dainty tentacles. When creating this type I reduced the spiral factor, which affects the way the tentacles bend and twist. I will not be using this type of tentacle.

Type # 4 – Creation Settings (Fig. 25)


Tubes
✓ Tubes
✓ Tube Completion
Creation

Tubes Per Step	0.500
Tube Rand	1.000
Start Tubes	0
Segments	100
Length Min	0.000
Length Max	4.000
Tube Width 1	0.003
Tube Width 2	0.001
Width Rand	0.000
Width Bias	0.000
Segment Length Bias	0.000
Segment Width Bias	0.000

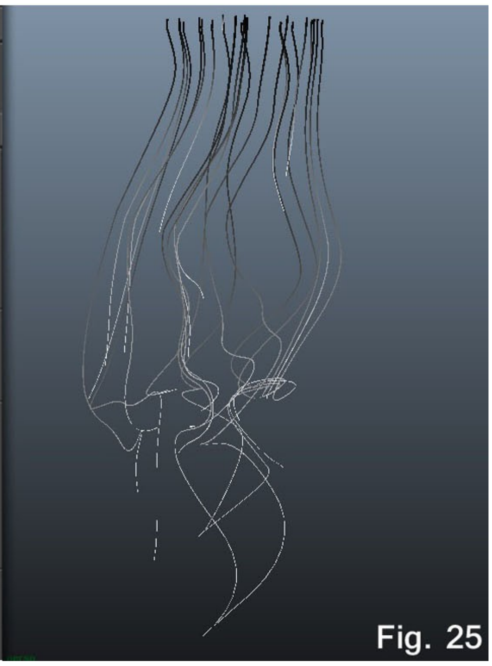


Fig. 25

Analysis
I used a small Step for this type as I felt a sparse population of these very thin tentacles could work well with Type #1 to fill out the inside of my Jellyfish Hood. A combination of the different styles will help enrich the overall look. I will be using this type in my tutorial

(c) Method #3 Using Ik-spline tool [15]

-EP curve (spline) to control tentacle movement.

What is IK spline tool?

Spline ik is a type of ik control which uses a spline to control a joint structure. It can be added to the joint chain when the joints are created or it can be added afterwards. They are normally used for tails, snake rigs, stretchable character back bone, or for what I need, tentacles.

Method – for building single tentacle

- First I created a proxy mesh of my tentacle for testing
- Secondly I created a joints chain which ran along the length of the tentacle
- Next I created an EP curve, using the (hold V) cmd to snap each control point to the joints, so the curve had as many points as the skeleton has. This is important for later in the process as the curve will be controlling our joints which in turn will drive the polygon deformation.
- With the three main objects created, I then used Maya Dynamics Hair tool to “Make selected curves dynamic” on 'curve1'. This creates an output curve 'curve2' which is attached to curve1 at both ends. Curve2 is the representation of curve1 after Hair dynamics have been calculated. I only needed the curve to be attached at the beginning of the joint, where the jellyfish head will be, so under follicleShape1, I changed 'Point Lock' to 'Base'. I also affected the stiffness scale of the curve to help create the floppy tentacle effect.
- Now we have the simulated curve, we need to attach it to the skeleton, we do this using the spline ik tool. With 'auto create curve' switched off, I selected the root of the skeleton then the tip, then shift-selected the output curve, 'curve2'. I did not want 'auto create curve' switched on because I had already create my own curve, which matches the skeleton. Now when curve2 moves, the skeleton will follow.
- All that's left to do is attach the mesh to the joints. I did this using Skin->Bind Skin->Smooth Bind. Now if we animate curve1, this cause curve2 to move and react dynamically, causing the joints to react identically, which in turn moves the mesh to create the final effect.

What I found when I attached it to my Test Rig

I attached several of the ikSpline tentacle to my Test Rig, and I noticed a few things about the movement which are less obvious when the tentacles are viewed alone. The ikSplines tend to behave in a pretty predictable manner, which can be useful in some cases, but when there are

15 **E. Keller** Maya Visual Effects: *The innovator's Guide*, p92-99

10 tentacles all repeating identical movements it breaks the illusion of reality. The tentacles tend to look quite heavy compared to the delicate Paint Effects strands I had tried out before. One advantage of this approach over nCloth and Paint Effects is that even after animating everything I can attach a new mesh at any point if I want more detail. With nCloth I would have to re-make the nCloth object and re-make all the solver settings each time I changed the mesh and Id have to re-draw the paint effects if I didn't like the movement. The type of movement achieved is similar to that which is seen on larger tentacles.

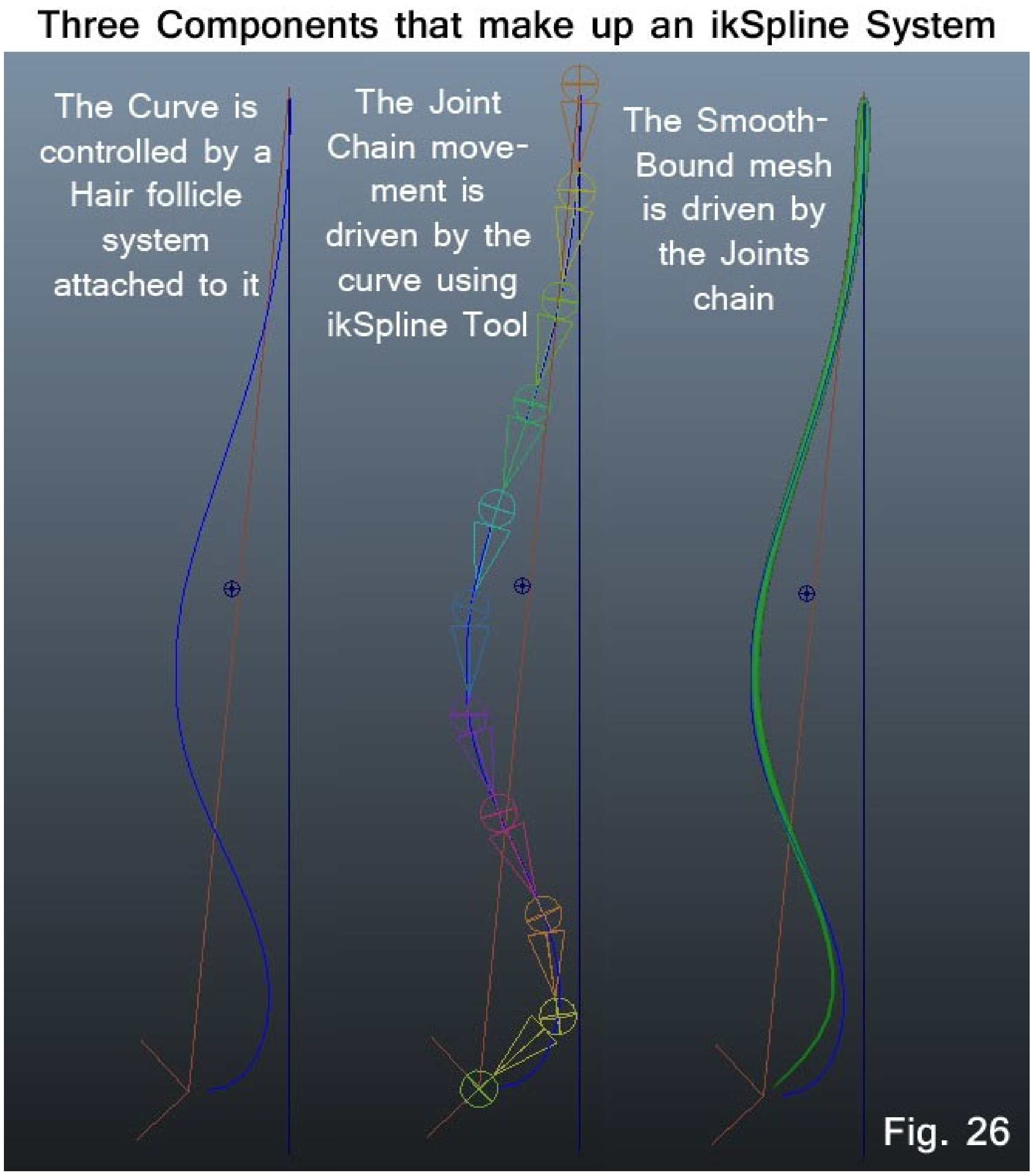
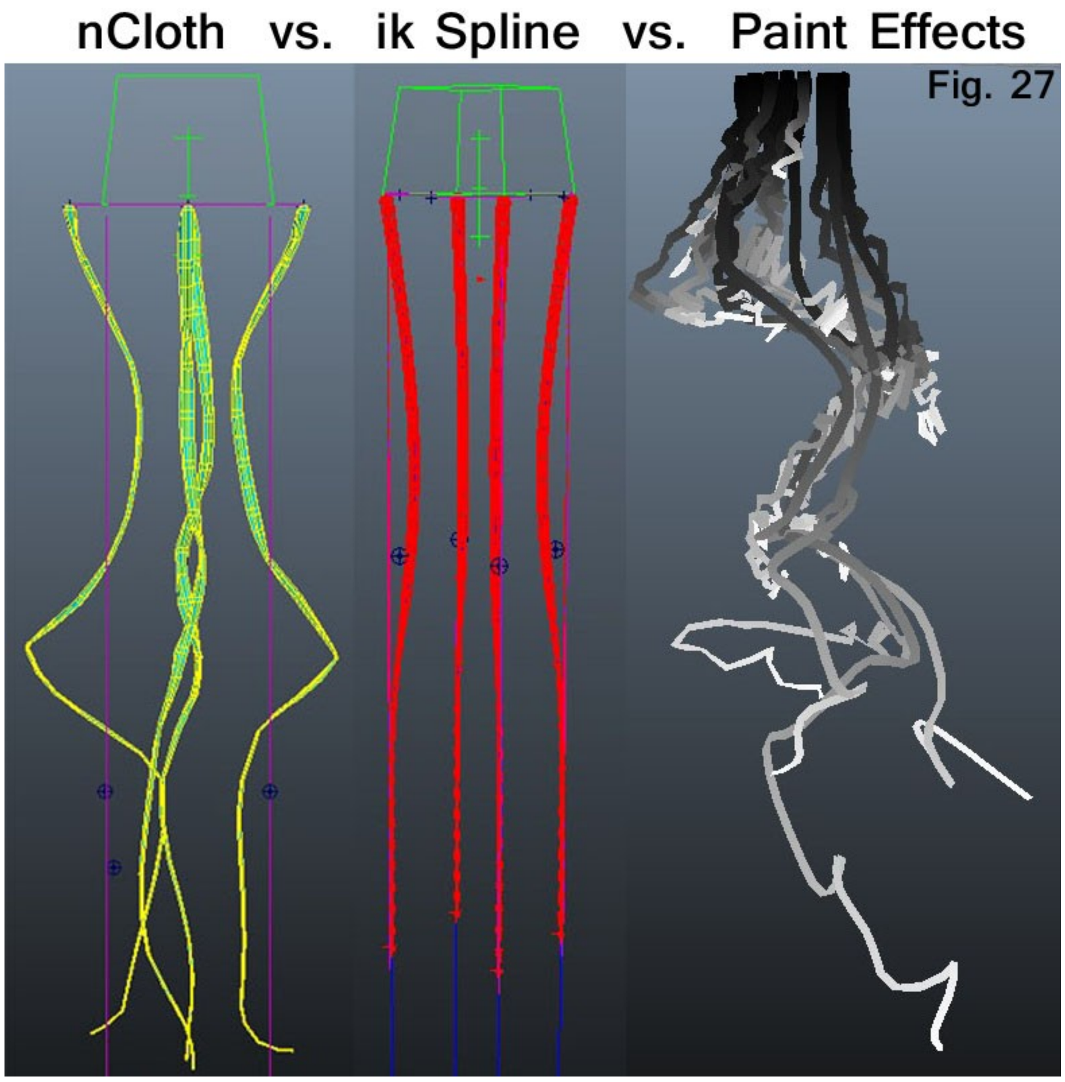

Three Components that make up an ikSpline System
The Curve is controlled by a Hair follicle system attached to it
The Joint Chain movement is driven by the curve using ikSpline Tool
The Smooth-Bound mesh is driven by the Joints chain

Fig. 26



Verdict

The ikSpline Method originally attracted me because I liked the fluid and predictable curves created down the spine, they were how I imagined tentacles to look. With the nCloth my initial reaction was that they lacked a sense of purpose or reason, it was like they are simply being dragged about. The ikSpline tentacles behaved more like muscular limbs, generating real kinetic forces to propel the Jellyfish. The unpredictable nature of nCloth was also a cause for concern because I wanted to know exactly how my tentacles were going to behave in any situation, without having to re-cache and re-animate to achieve the desired nCloth look. It was only when I came to attaching my final Hood model with the rig, and rendered out 200 frames of movement that I really noticed the potential of using nCloth. Although I liked the more deliberate movement of the ikSpline, the nCloth broke up the repetitiveness of so many tentacles, and after playing with a few settings I was able to get some very convincing results. For my tutorial I will include both ikSpline and nCloth as options. This is because I feel that it should be up to the artist to decide which works best for them and their particular breed of Jellyfish. Using Paint Effects to fill-out the centre of my Jelly, I had created all aspects required to make a 3D Jellyfish.

6) Tutorial

(working units set to cm)
key: LMB- left mouse button
RMB- right mouse button

(i) Modelling the Jellyfish Hood.

1. Make a new scene file with **file->new scene**
2. Switch to the front view with **panels->Orthographic->front**
3. Switch on *snap to grids* tool in the snap icons tray

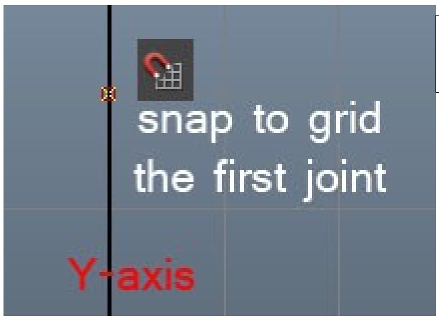


Fig. 01

4. Now we want to make an *EPcurve* so we can draw our Jellyfish Hood. Do this by selecting **create->EP curve Tool *with the options box* .** In the options box, make sure you have a curve degree of *3 cubic* so that we get a smooth line, and ensure *'uniform knot spacing'* is switched on. I based the Jellyfish body on an Pacific Sea Nettle, using reference images to get the shape for my curve. The first and last joint you make must be grid-snapped on to the centre of the y-axis (Fig.01). When you make your first joint, click anywhere along the Y-axis.

5. Then **deselect snap to grids,** before drawing your hood shape, in an area the size of roughly 4 units by 4 units. (Fig 02)

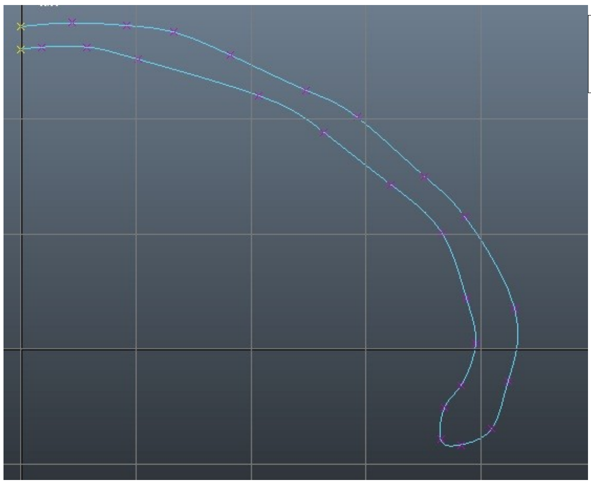
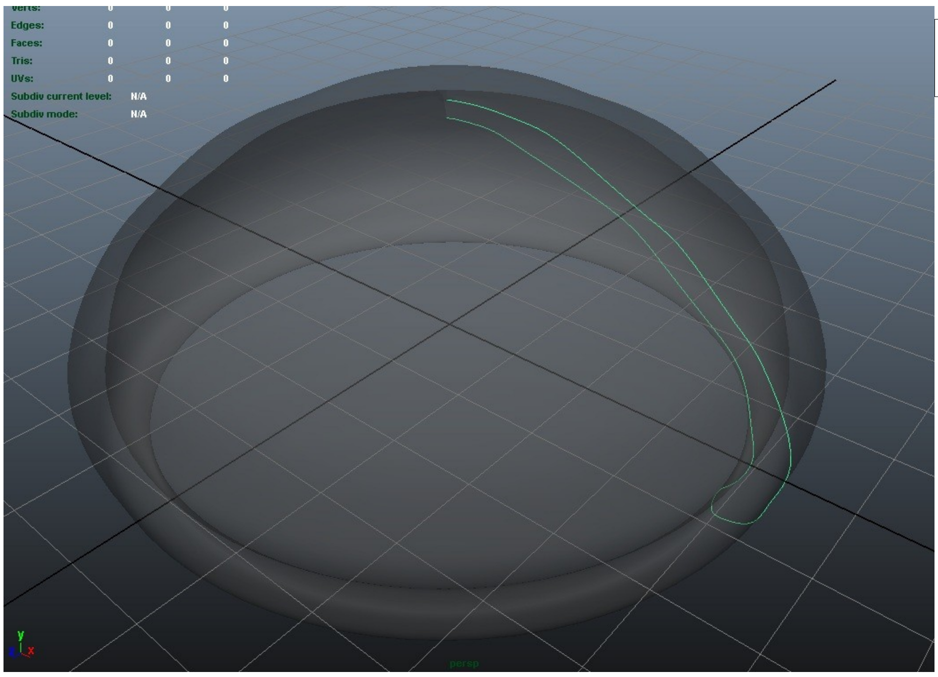


Fig. 02

Once the Hood shape is complete, **reselect snap to grids** and make the last point snap to the y-axis.

6. With the EP curve created, select the curve1 then use **Surfaces->Revolve Tool with the options box** (Fig 03). Reset the tool to the default settings then change the number of segments to 20 then click revolve.



```
Verts:              0    0    0
Edges:              0    0    0
Faces:              0    0    0
Tris:               0    0    0
UVs:                0    0    0
Subdiv current level:  N/A
Subdiv mode:           N/A
```

Fig. 03

7. This creates a *Nurbs surface* object called **revolvedSurface1** which we can convert to polygons. Select **revolvedSurface1** then use **Modify->Convert Nurbs To Polygons** reset the tool to default and change Type to **quads** (Fig 04) **.**

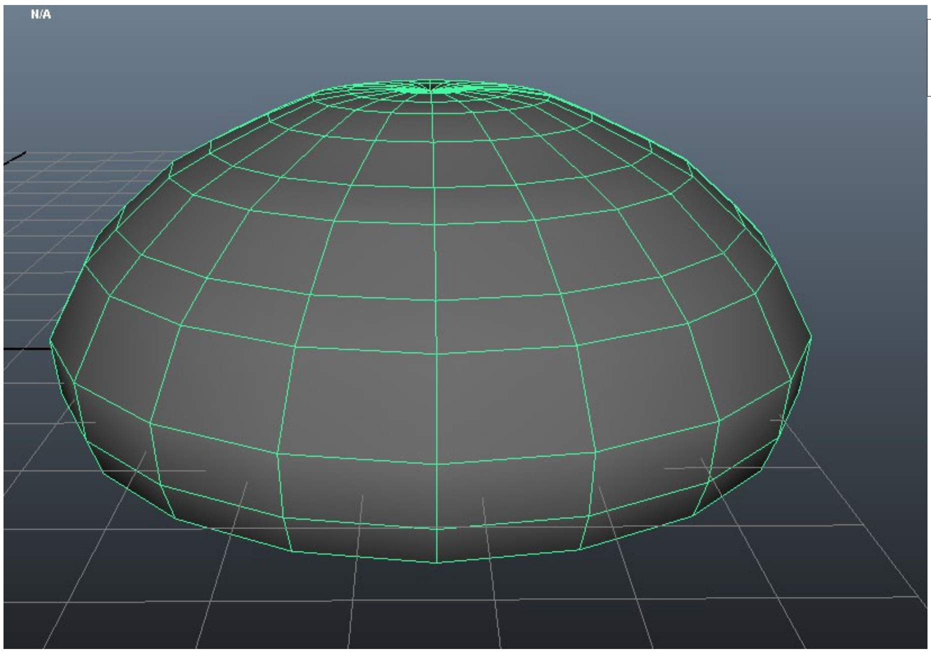


Fig. 04

8. Now we have our polygon Model we no longer need the Nurbs surface, so delete the Nurbs mesh. Before you continue, there are gaps at the centre of the mesh from revolving which could cause problems later. Solve this by selecting all the *vertices* in the centre of the mesh with **select->vertices** then **LMB** drag over all the vertices in the centre. Although it looks like one vertices, there is actually one for every line going into the centre. This occurs both where our curve begun and where it ended (Fig 05).

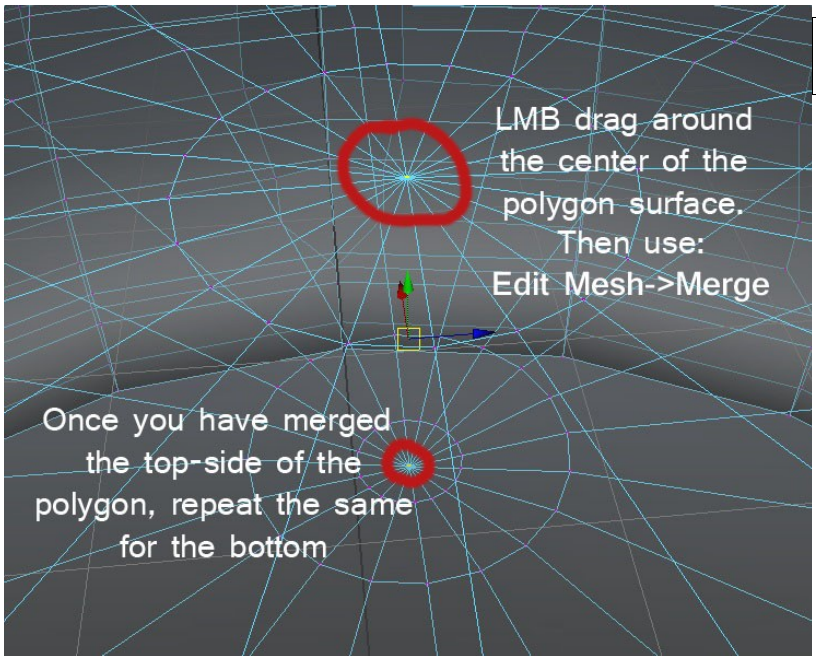

LMB drag around the center of the polygon surface. Then use: Edit Mesh->Merge
Once you have merged the top-side of the polygon, repeat the same for the bottom

Fig. 05

9. To merge the vertices use **Polygons->Edit Mesh->Merge Tool**. Do this for both the inside and outside centre point

10. Pacific Sea Nettles have ridges running down the side of their bodies, adding simple details like this will enhance the look of the hood mesh. We can do this very simply by extruding some faces to create the ridges.

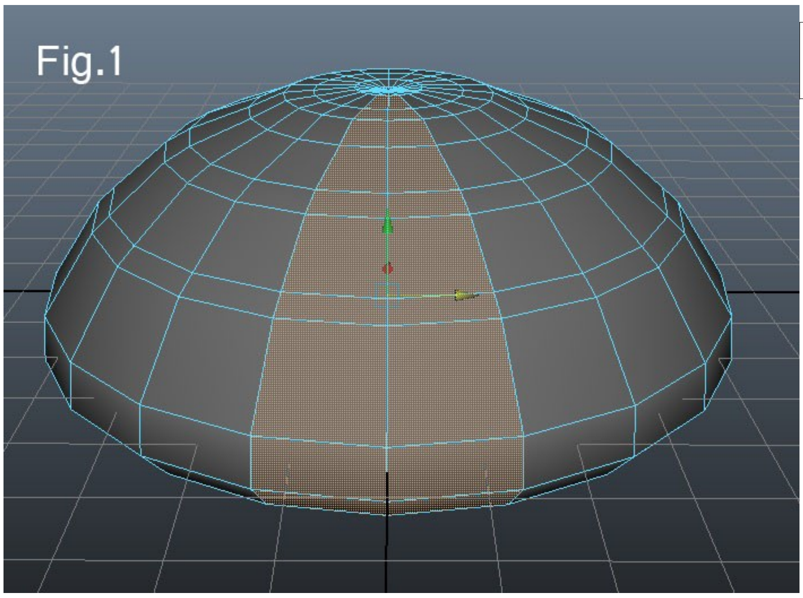
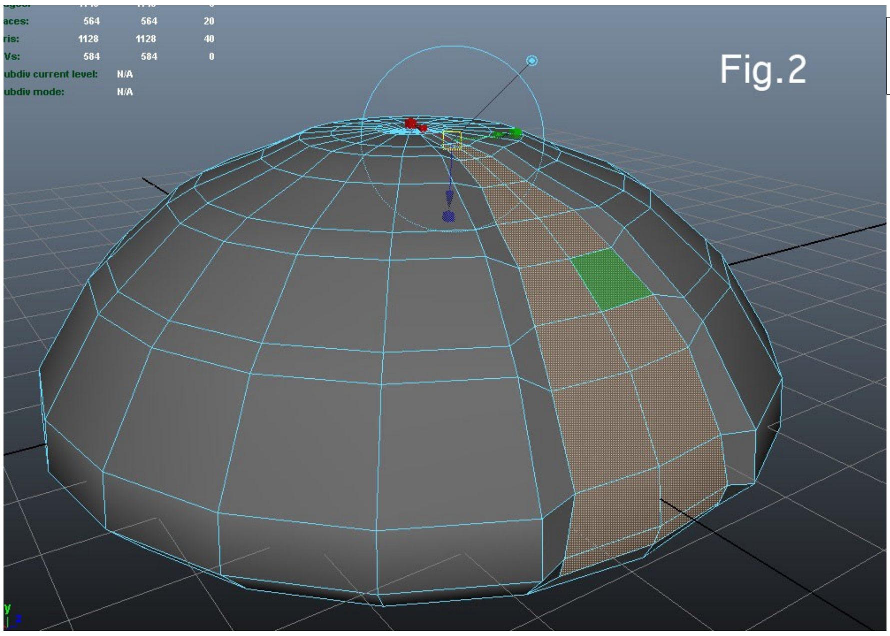


Fig. 06

11. Select two sets of faces down one side of your mesh as shown (Fig.6), and go to **Polygons->Edit Mesh->Extrude *with options box*.** Reset the Tool and then set the *offset to* **0.5** (Fig.7). This means we don't extrude exactly out from the faces, the offset gives us a smaller extrude area within the selected faces.



```
aces:           564      564     20
ris:           1128     1128     40
Vs:             584      584      0
ubdiv current level:   N/A
ubdiv mode:            N/A
```

Fig. 07

12. Inside the mesh *NurbsToPoly1* attributes go to **polyExtrudeFace1->Poly Extrude Face History** then set the *local Z-translate* to **-0.3.** This will then extrude very subtly into the mesh. This needs to be repeated around the geometry with identical settings until the entire mesh has extrusions all the way around (Fig. 08).

13. The finished geometry should look something like (Fig.8). With the basic geometry complete, we need to create the UV's so that it can be textured. To do this, select the *NurbsToPoly1* mesh, then go **Polygons->createUVs->Automatic Mapping**. This produces a pretty procedural and basic UV map, but it works for what we need. If you wish for a more accurate UV map you could use planar mapping and unwrap the model manually.

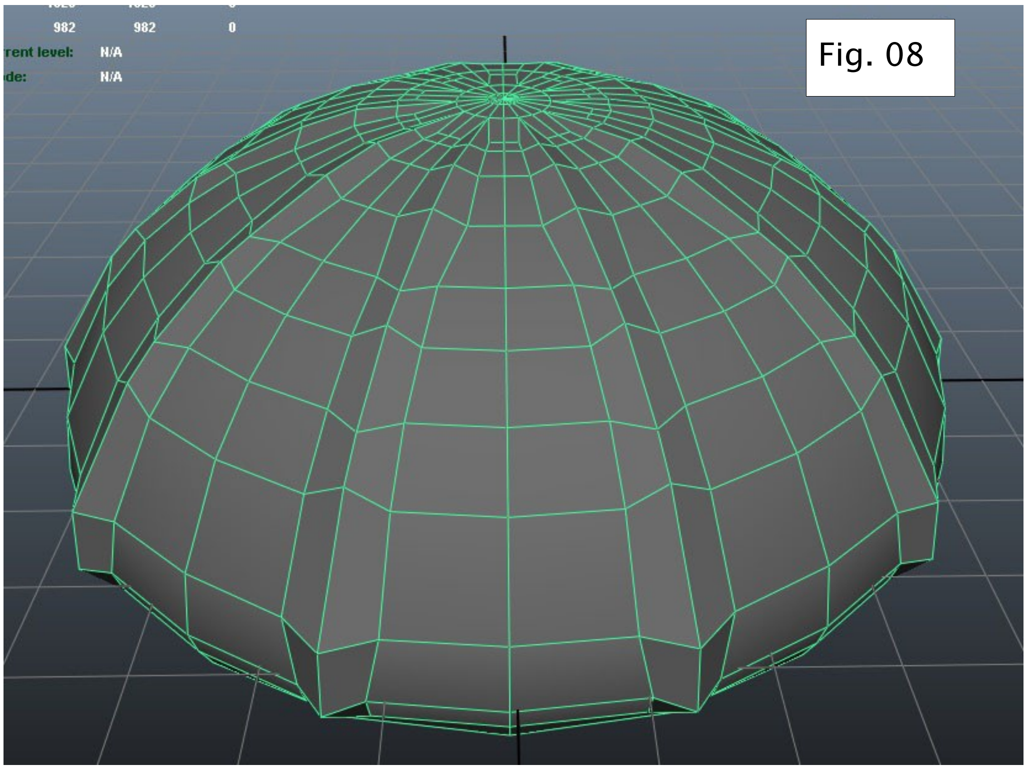
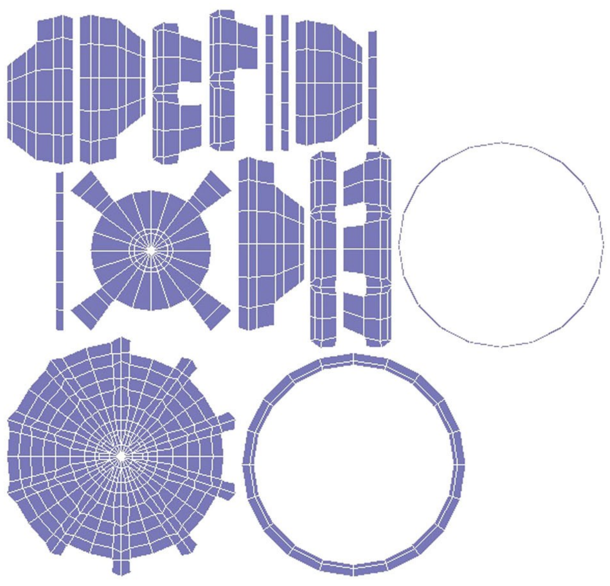


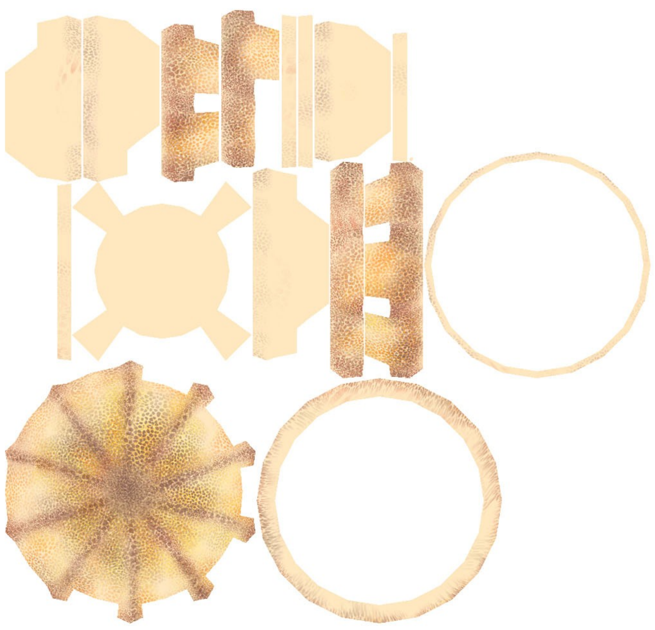
Fig. 08

14. Now you can take a UV-snapshot to be used for texturing. Inside the UV-Editor click on **Subdivs->UVSnapshot** (Fig 09) and create your own diffuse in photoshop. Alternatively, you can simply apply a solid colour to the mesh by **RMB** over the *NurbsToPoly1* mesh and clicking **Assign new Material->Lambert** then set the diffuse to whichever solid colour you wish to use. If you wanted to play with the Transparency of the texture, you could

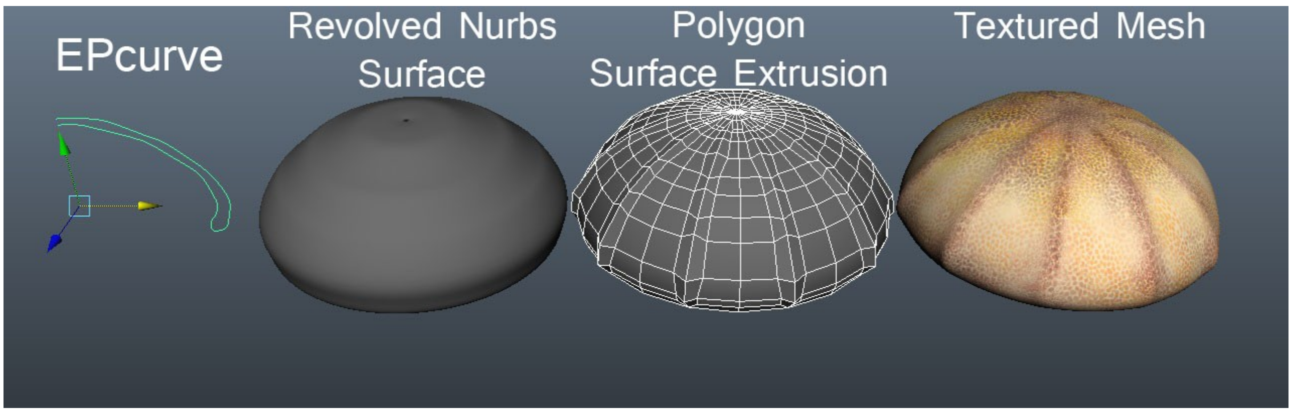
15. Once the texture file has been created, **RMB** over the *NurbsToPoly1* mesh then click **Assign new Material->Lambert.** Click the Colour node's **checkered options box**, and attach a *file node*. Then browse to your image file to attach the new diffuse map.

Fig. 09





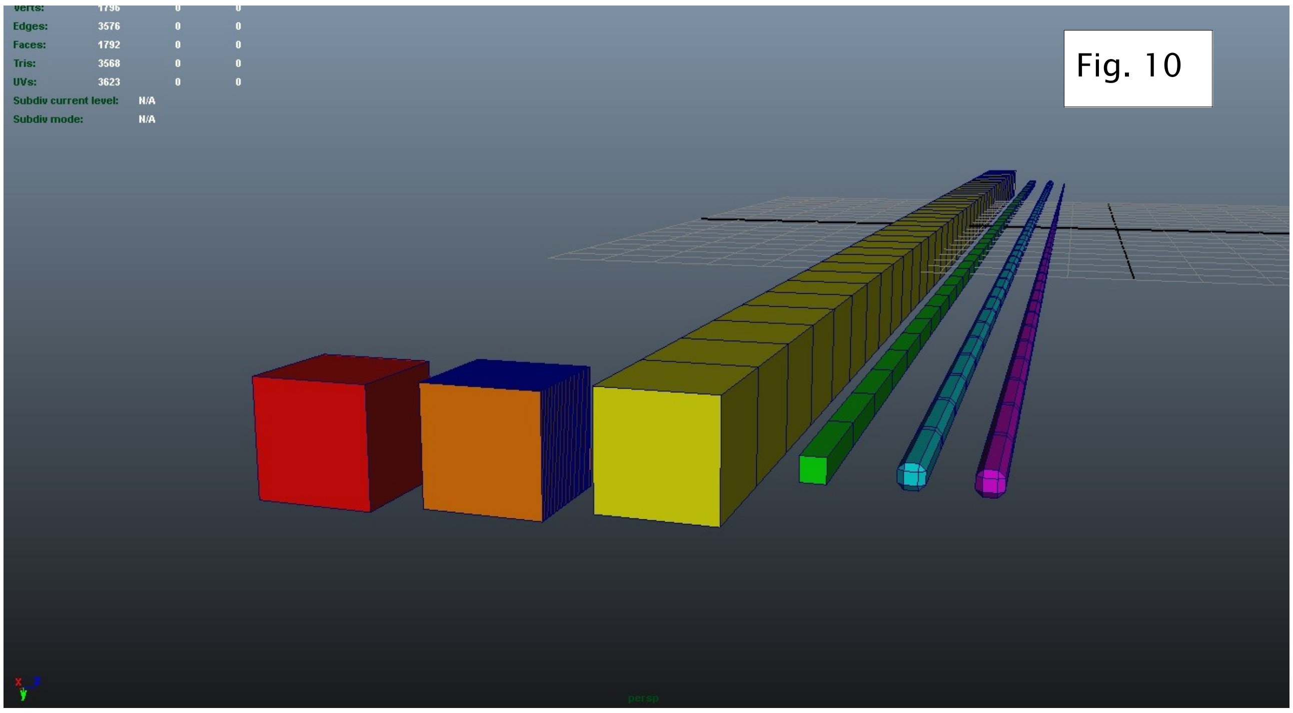
16. Once Textured use **Edit->delete by type->history** followed by **Modify->Freeze Transformations.** The model is now ready to be used in a Jellyfish rig.
17. Save the scene so that we can import it later on when we set-up our Jellyfish.



Jack Kersey i7827320

(ii) Modelling a simple large tentacle for nCloth or ikSpline

1. Make a new scene file with **file->new scene**
2. Create a polygon Cube with **create->Polygon Primitives->*Cube***
   -note: turn off the 'interactive creation' tick box
3. In the **pCube1** attributes, under the **polyCube1** tab, change the Subdivisions Height to 40.
4. Now under **pCube1** Transform Attributes change the Y axis Scale value to 45 and change the X and Z scale to 0.2
5. Now RMB over the mesh, **select->Edges,** and **LMB** Highlight all the edges.
6. Then use ***EditMesh->Bevel*** with the options box. Change the Offset to 0.5.
7. Now select the Geometry,choose **CreateDeformers->Non-linear->Flare.** Move the ***flare1Handle*** to 22.5 in the Y-axis (the same value as the deformer's XYZ scale). Inside *flare1* attributes, change the *Low Bound* to -10 and the **High Bound** to 0. Now make the **Curve** value -1.5.
8. Select the mesh and use Hot Keys **Ctrl-D** to duplicate the deformed geometry, then delete everything else in the scene. Rename the duplicated mesh 'tentacle_mesh' and before saving the scene file with the same name, select the mesh then use **Modify->Freeze Transformations.** (see Fig. 10 for coloured stages).
9. Finally attach a simple Lambert shader **RMB->Assign new material** over the mesh and make it a dark shade of Ocre.
10. Save the scene so that we can import it later on when we set-up our ikSpline and nCloth.

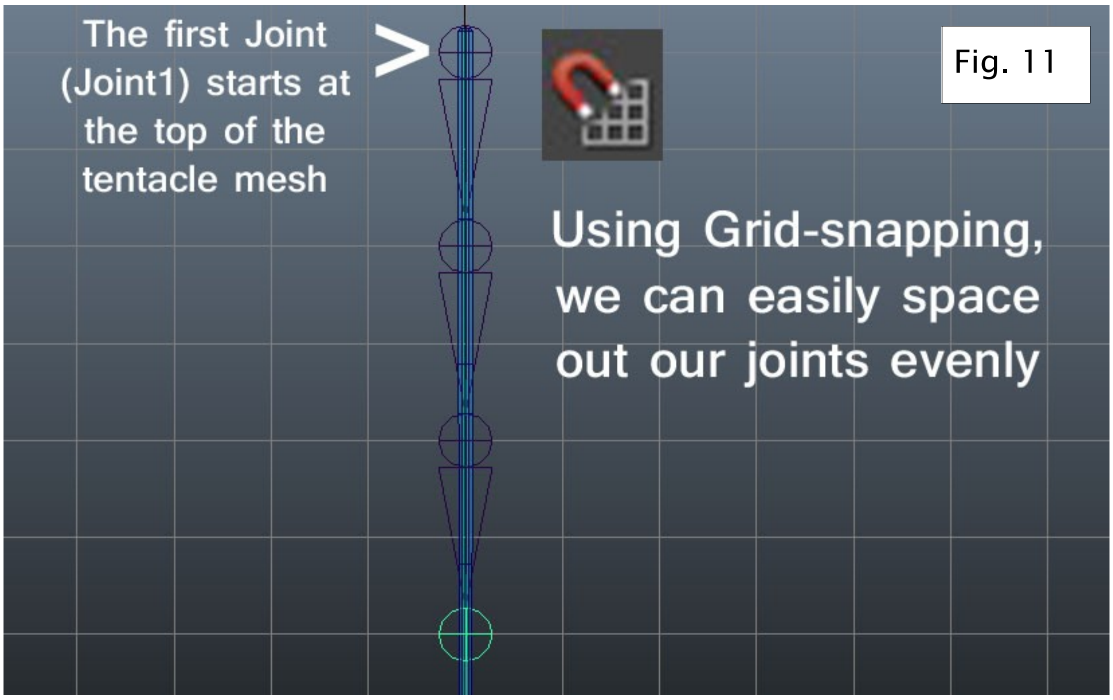


```
Verts:     1796    0    0
Edges:     3576    0    0
Faces:     1792    0    0
Tris:      3568    0    0
UVs:       3623    0    0
Subdiv current level:   N/A
Subdiv mode:            N/A
```

Fig. 10

(iii) Rigging large tentacles with ikSpline Tool

1. Open a new scene with **File->New Scene**
2. Import your tentacle geometry which is to be rigged, name it 'tentacle mesh'
3. Select the **Animation** menu set
4. Firstly we want to create a Joints chain which runs down the entire length of the mesh. Use **Skeleton->Joint Tool** with grid snapping turned on. Then make joints all the way down the geometry in a straight line at intervals of three units. Using grid snapping allows you to create a straight line with uniform spacing between each joint (Fig 11).


The first Joint (Joint1) starts at the top of the tentacle mesh
>
Using Grid-snapping, we can easily space out our joints evenly
Fig. 11

5. Now we want to create an EP-curve **create->EPcurve Tool.** Make sure **'3 cubic'** is selected in the tool options (Fig. 12). This curve will be the spline that controls the movement of our tentacle.

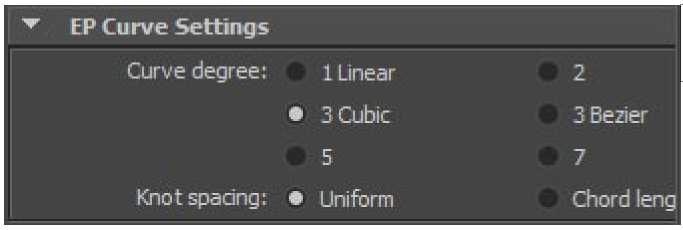

EP Curve Settings
Curve degree: ○ 1 Linear ○ 2 [✓ 3 Cubic] ○ 3 Bezier ○ 5 ○ 7
Knot spacing: [✓ Uniform] ○ Chord leng

Fig. 12

6. In the Front view, whilst you have the EPcurve tool selected, use Hot key **V** (Fig 13) and move the mouse over the root **Joint1** of the chain. Now click to create the first point on the curve. Hold **V** again and do the same on the next Joint in the chain until you reach the end, then hit **Enter** to finish.

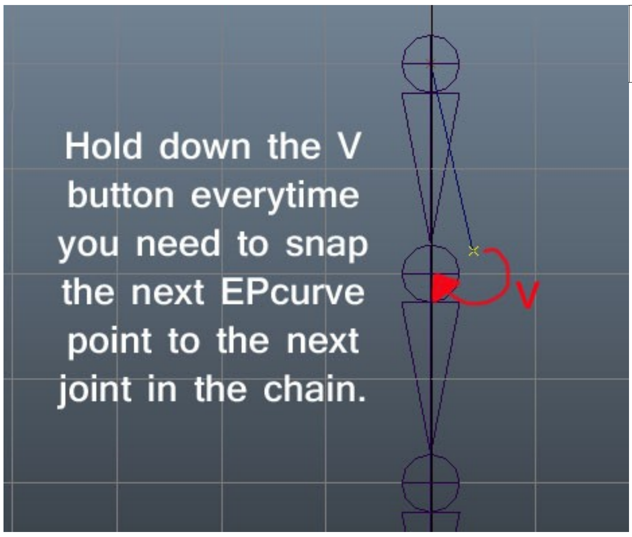
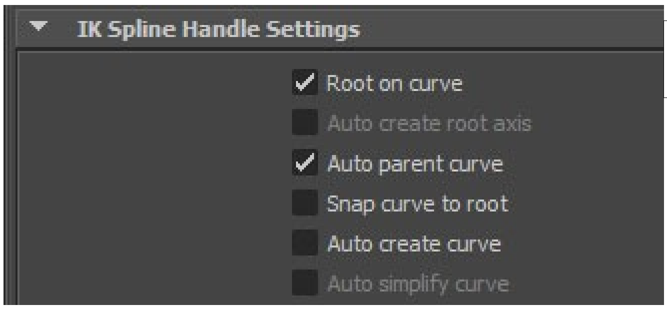

Hold down the V button everytime you need to snap the next EPcurve point to the next joint in the chain.
V

Fig. 13

7. Select the **nDynamics** menu set
8. We now have a curve which follows the exact position of the joint chain in world space. We need to make the curve dynamic, we do this by selecting **curve1** , then use **Hair->Make Selected Curves Dynamic.** This creates a **hairSystem** with a **follicle1** and an Output **curve2.**
9. Now inside **follicle1** attributes change **PointLock** from **'BothEnds'** to **'Base'.** This means our curve can move freely from the base downward like a tentacle does.

10. Now to attach the joints to our dynamic curve. This is done using the ikSpline tool. Inside the Animation menu set use **Skeleton->ikSpline Tool** with the options box. Inside the options box reset the tool settings, and make sure 'Auto create curve' (Fig 14) is switched off because we already have a curve target for our spline .


IK Spline Handle Settings

- [✓] Root on curve
- [ ] Auto create root axis
- [✓] Auto parent curve
- [ ] Snap curve to root
- [ ] Auto create curve
- [ ] Auto simplify curve

Fig. 14

With the ikSpline tool selected, click the **first** then the **last** Joint in the chain, then bring up the **Hypergraph.** You should see your Joint and mesh nodes in there along with the hair system which has two curves. You need to *shift-select* **curve2** (Fig 15). This will now connect **curve2** to our joints, making the joints move when the curve bends.

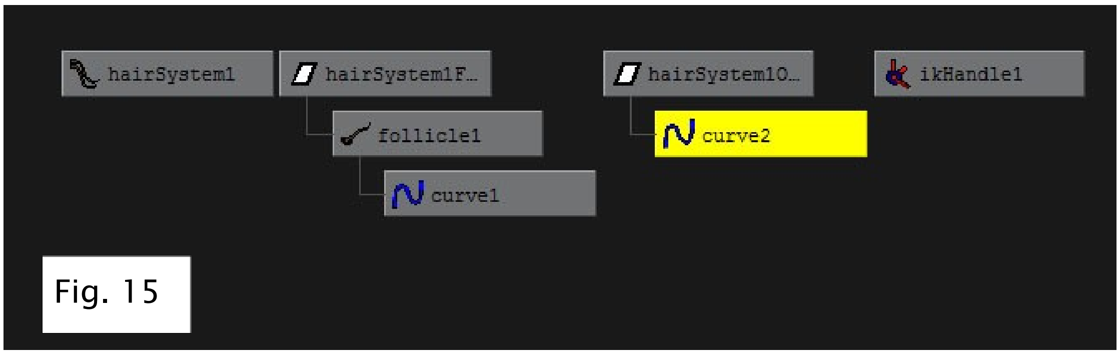
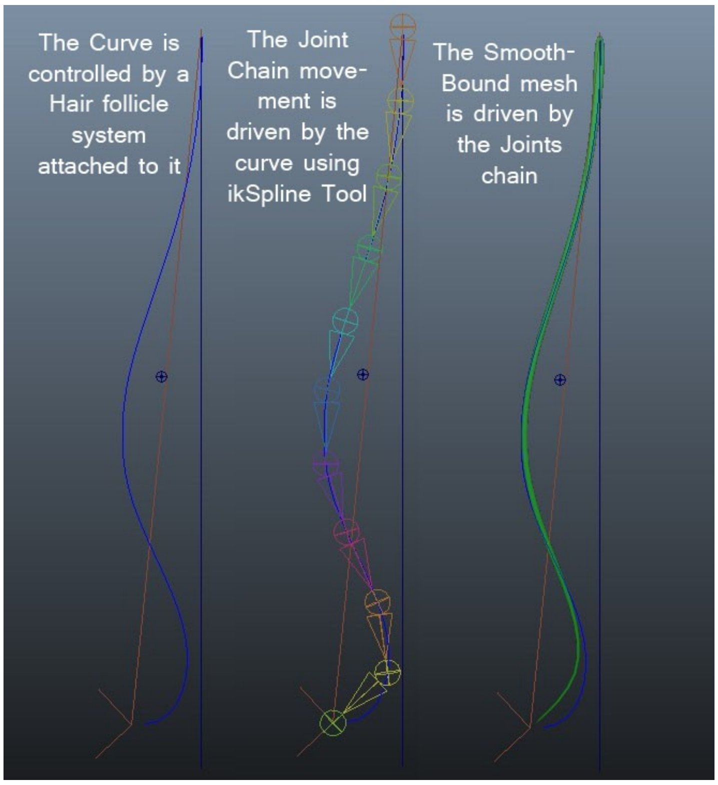


Fig. 15

11. Finally shift-select the tentacle mesh and the joints and use **Skin->BindSkin->Smooth Bind.** To test if the spline is working, animate **curve1** to move 10 places in the X between 0 and 10 frames. This should cause the **curve2** to bend and ripple, which in turn causes the **Joints** to move, which in turn makes the **mesh** move. When animating this set-up, only ever key-frame curve1 as this is at the top of the control hierarchy.


The Curve is controlled by a Hair follicle system attached to it
The Joint Chain movement is driven by the curve using ikSpline Tool
The Smooth-Bound mesh is driven by the Joints chain

12. Save the scene so that we can import it later on when we set-up our Jellyfish.

What you should have in your scene: A joint chain, a hairSystem1, follicle1, curve1, curve2 and an ikHandle1 (Fig. 16)

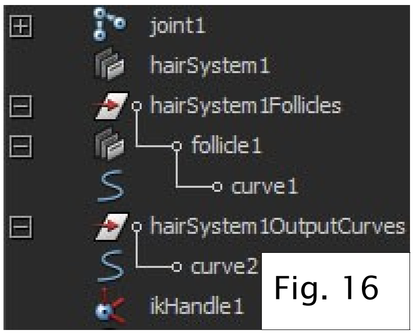


Fig. 16

(iv) Rigging large tentacles with nCloth

1. Open a new scene with ***File->New Scene***
2. Import your tentacle geometry which is to be rigged , name it 'tentacle mesh'
3. Select the **nDynamics** menu set
4. Select your imported *tentacle mesh* and click **nMesh->create nCloth**
5. This should create **nClothShape1.** If you press play on the time line, the nCloth will simply drop down into infinity. We can stop this by using a simple Dynamics constraint
6. Select the top few *vertices* (Fig) of the mesh by holding down **right click->vertices** then using **LMB-drag** (Fig. 17) to select all the vertices you want.

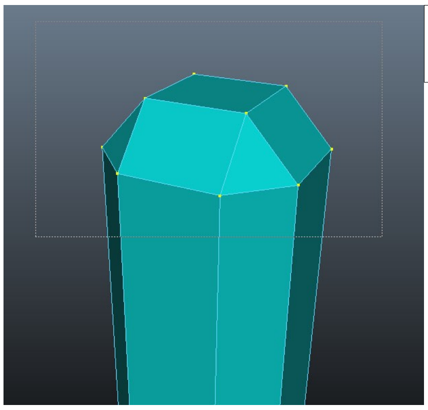


Fig. 17

7. Then when your happy with the vertices you have selected, click **nConstraint->Transform** This creates a constraint that suspends the selected vertices in world space, and prevents the nCloth from pulling the mesh down. The mesh now effectively hangs off these vertices like a coat on a hanger.
8. Inside the Attributes of **nClothShape1->Collisions** change Thickness to **0.035.**
9. Now add a turbulence field by selecting the nCloth mesh, then **fields->turbulence** and reset the tool to default.
10. Under Turbulence field attributes, change Magnitude to **50** and Attenuation to **0.**
11. This field helps to break up the look of the nCloth so that it has a more organic feel.
12. We have successfully created a simple nCloth tentacle. Save the scene so that we can import it later on when we set-up our Jellyfish.

What you should have in your scene: one mesh, one nCloth, one dynamic Constraint (Fig. 18);

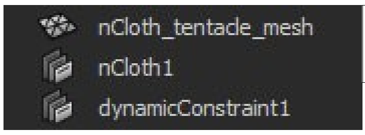

nCloth_tentacle_mesh
nCloth1
dynamicConstraint1

Fig. 18

(V) Making Paint Effects for smaller tentacles

1. Open a new scene with **File->New Scene**
2. Create a polygon Plane with **create->polygonPrimitives->Plane**
3. Scale *Plane1* by **10** in the XY and Z axis
4. Bring up the **Rendering** menu set with **F6**
5. Now select the mesh *Plane1* and use **Paint Effects->Make Paintable**
6. Reset the template brush with **Paint Effects->Reset Template Brush**
7. The click **Paint Effects->Template Brush Settings**
   use the settings in Fig 19 to set up your Paint Effects;

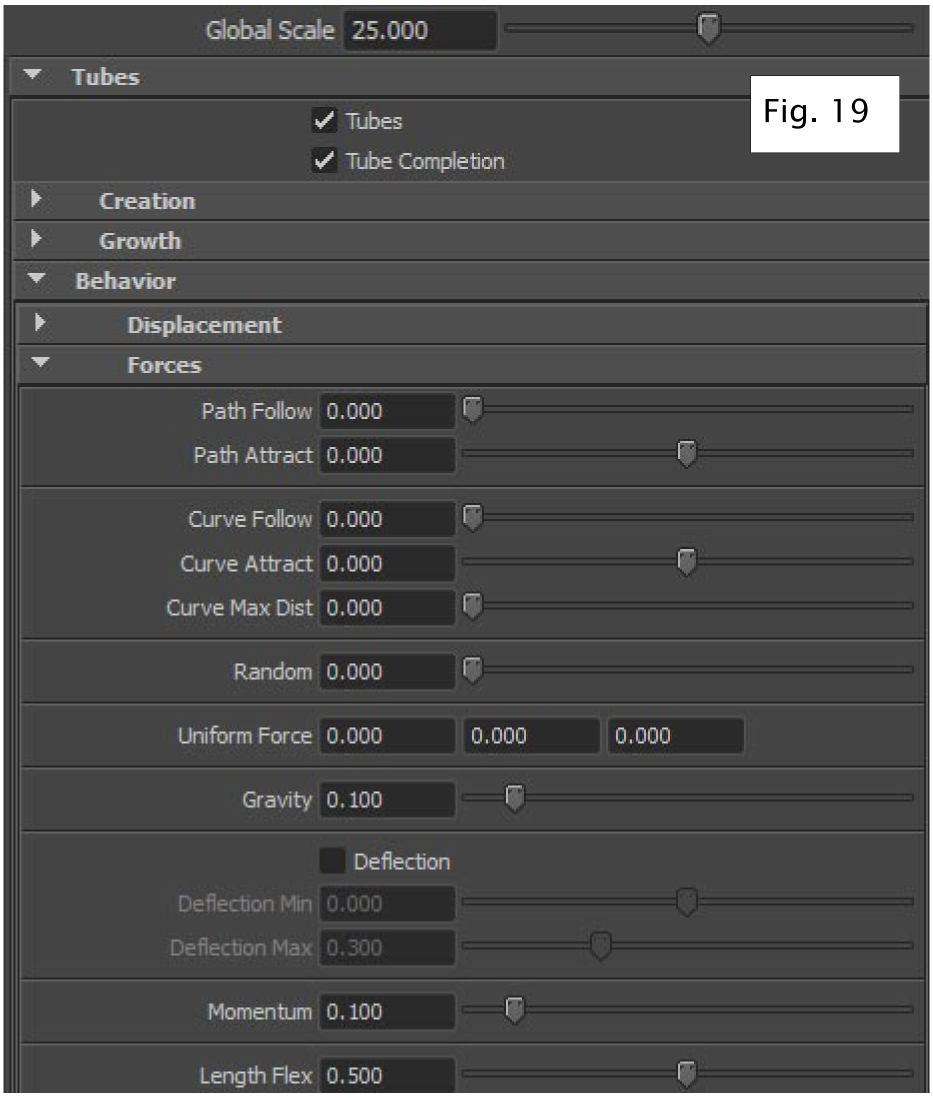


Fig. 19

Global Scale 25.000

Tubes
- ✓ Tubes
- ✓ Tube Completion

Creation

Growth

Behavior

Displacement

Forces

Path Follow	0.000
Path Attract	0.000
Curve Follow	0.000
Curve Attract	0.000
Curve Max Dist	0.000
Random	0.000
Uniform Force	0.000 0.000 0.000
Gravity	0.100
Deflection	☐
Deflection Min	0.000
Deflection Max	0.300
Momentum	0.100
Length Flex	0.500

8. Now we going to make two different styles of tentacle from these settings. Both types will share the behavioural attributes we have set, but will have different creation settings.

For Type_01(Fig 20):

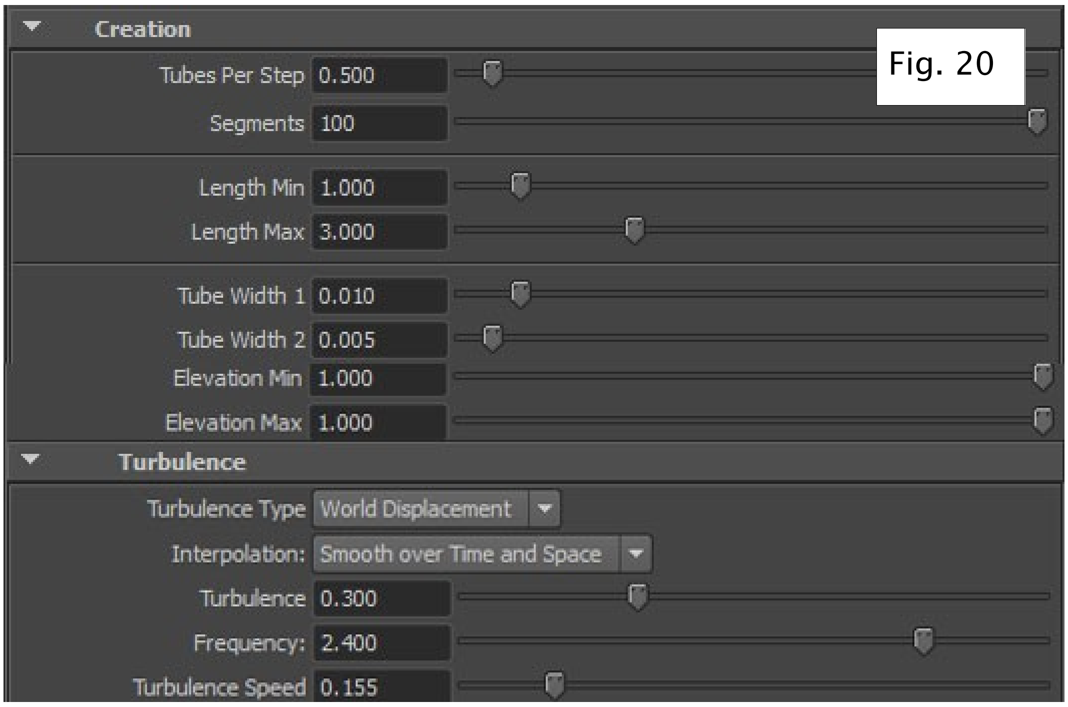


Fig. 20

Creation

Tubes Per Step	0.500
Segments	100
Length Min	1.000
Length Max	3.000
Tube Width 1	0.010
Tube Width 2	0.005
Elevation Min	1.000
Elevation Max	1.000

Turbulence

Turbulence Type	World Displacement
Interpolation:	Smooth over Time and Space
Turbulence	0.300
Frequency:	2.400
Turbulence Speed	0.155

Now paint a small circle onto the poly plane to draw the paint Effects

For Type_02 (Fig 21):

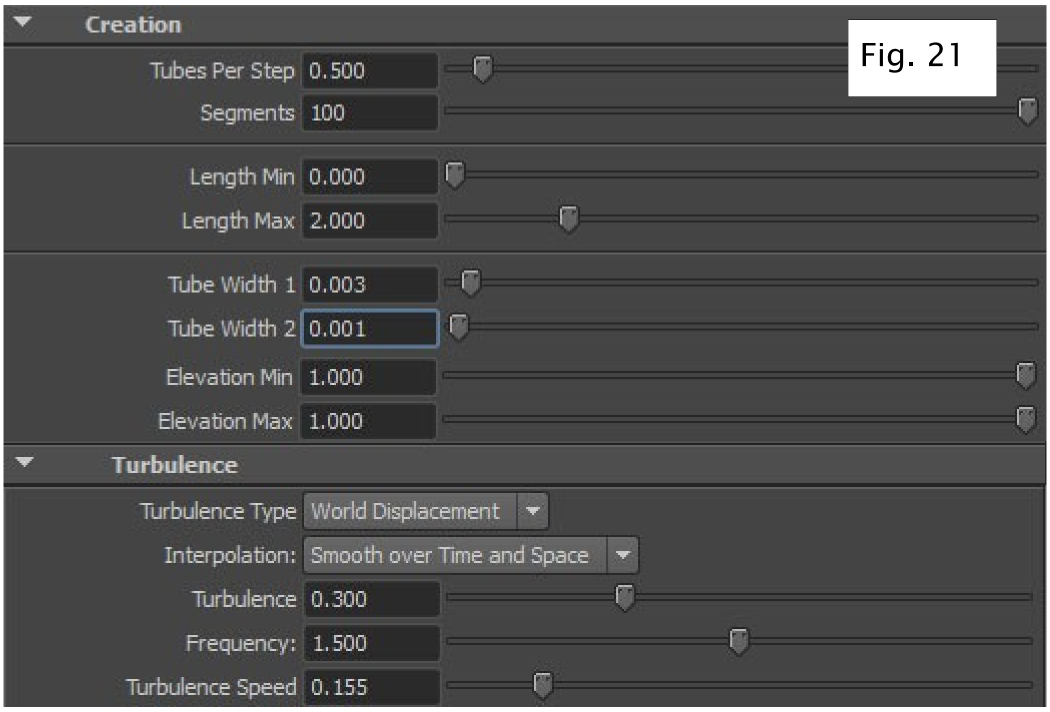


Fig. 21

Creation

Tubes Per Step	0.500
Segments	100
Length Min	0.000
Length Max	2.000
Tube Width 1	0.003
Tube Width 2	0.001
Elevation Min	1.000
Elevation Max	1.000

Turbulence

Turbulence Type	World Displacement
Interpolation:	Smooth over Time and Space
Turbulence	0.300
Frequency:	1.500
Turbulence Speed	0.155

Now paint a small circle onto the poly plane to draw the paint Effects

8. The final stage of making these tentacles is converting them to polygons for rendering.

9. Select both paint effects objects, *strokeBrush1* and *strokeBrush2,* then click **Modify-> Convert->Paint Effects to Polygons** with default settings.

10. Do not delete the paint effects, because without them, the converted polygons will not be dynamic.

11. Because we have saved the two types of brushes as presets, if we ever want to draw more of the same tentacles we can simply grab the pre-set brushes in the workspace toolbar and draw away without having to re-do all the settings.

12. Save the scene so that we can import it later on when we set-up our Jellyfish.

What you should have in your scene: two brush strokes, two converted meshes, and a polygon plane (Fig 22);

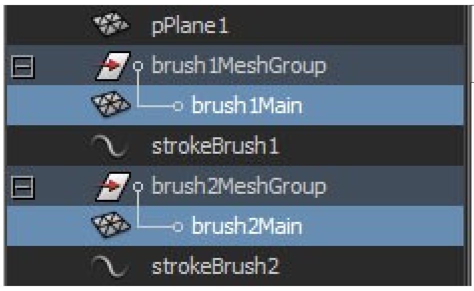


Fig. 22

We will use both types of Paint Effects Tentacles in the final Composition (Fig 23):

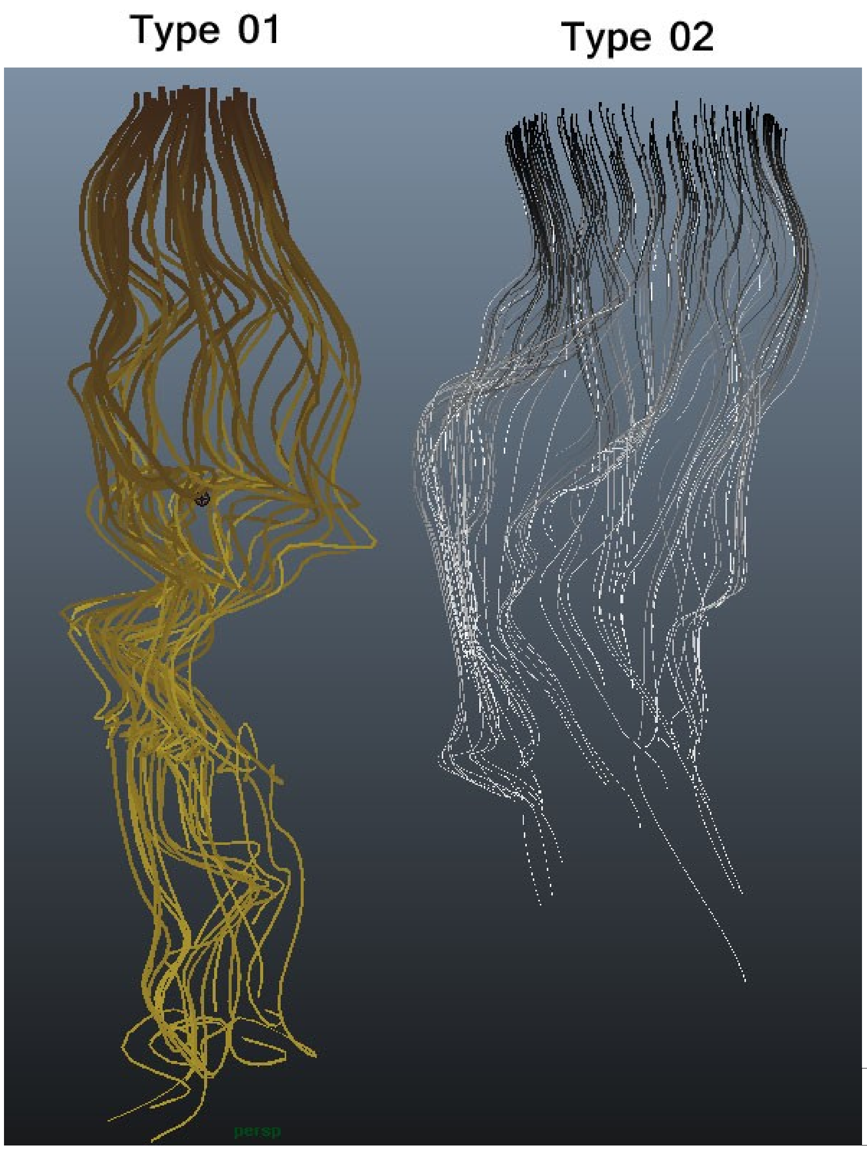
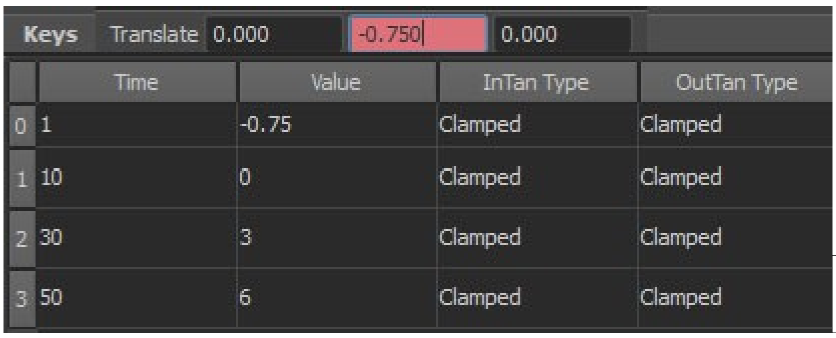


Fig. 23

(vi) Rigging and Animating the Body

1. Open a new scene with **File->New Scene.**
2. Import the Jellyfish Hood polygon mesh with **File->Import.**
3. We want to animate the Hood mesh movement in the y-axis first, use the following settings in Fig. 24.

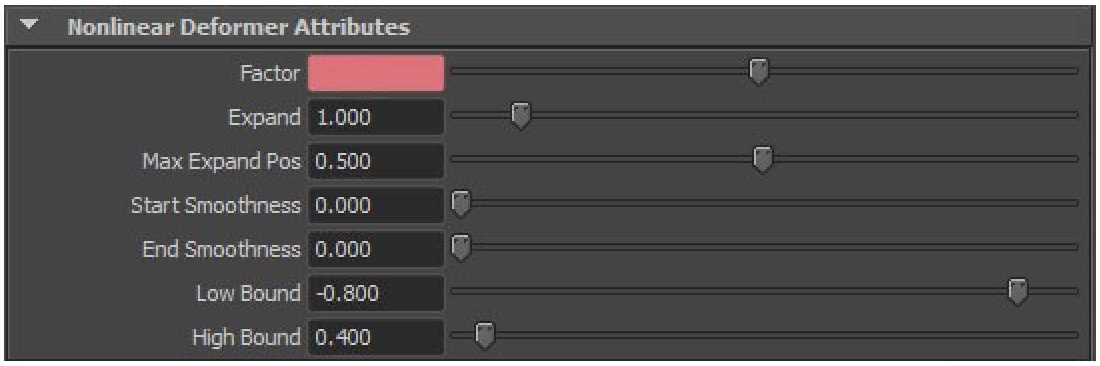


Keys Translate 0.000 -0.750 0.000

	Time	Value	InTan Type	OutTan Type
0	1	-0.75	Clamped	Clamped
1	10	0	Clamped	Clamped
2	30	3	Clamped	Clamped
3	50	6	Clamped	Clamped

Fig. 24

4. Now click **Window->Animation Editors->Graph Editor**
5. With the mesh translate-Y curve selected, click **Curves->Post Infinity->Cycle with Offset** this will mean the Y-translation displacement of our mesh is repeated after the final frame (50).
6. Now we need to create two squash Handles, one for the Main Hood mesh, and another to squash the Hood margin. The following is the settings we want for both deformers, as you can see *Factor* is in Red (Fig. 25) because we are going to key frame that separately for each deformer:



Nonlinear Deformer Attributes

Factor	🟥
Expand	1.000
Max Expand Pos	0.500
Start Smoothness	0.000
End Smoothness	0.000
Low Bound	-0.800
High Bound	0.400

Fig. 25

7. Set the First Squash handle deformer as follows:
   select the mesh, **Create Deformers->non Linear->squash**

   Once you have all the set Attributes up (Fig. 25), move the squash deformer to about the middle of the Hood mesh (Fig. 26).

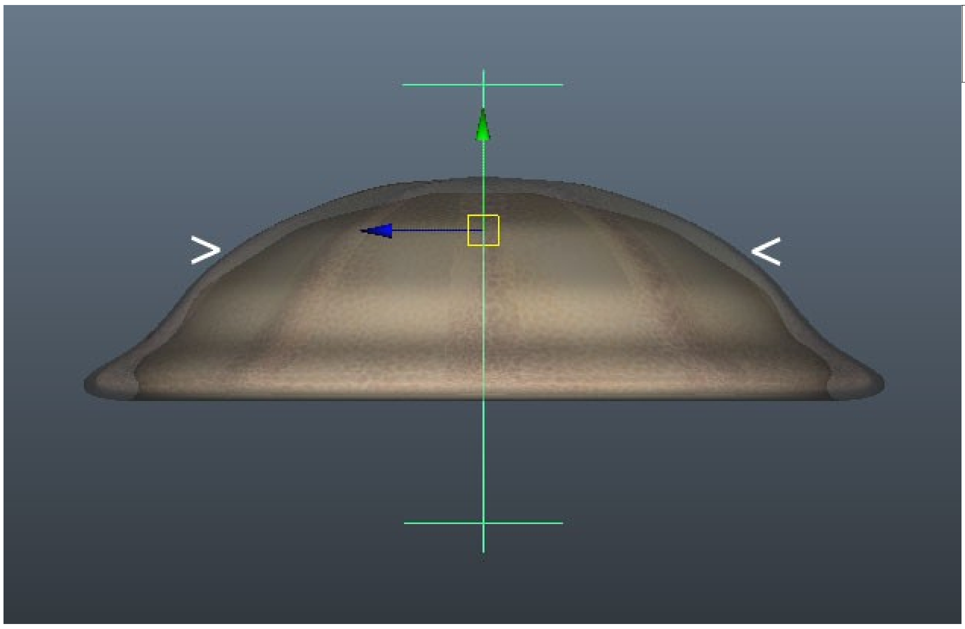


Fig. 26

8. And key-frame the *Factor*, using **RMB-click** over *Factor* then **click->Set Key** on the frames in Fig. 27.

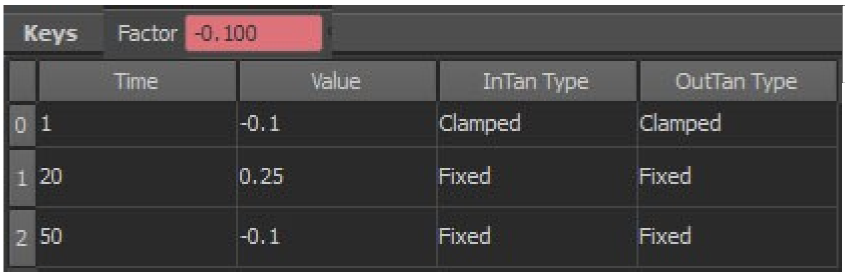


Fig. 27

Keys | Factor -0.100

	Time	Value	InTan Type	OutTan Type
0	1	-0.1	Clamped	Clamped
1	20	0.25	Fixed	Fixed
2	50	-0.1	Fixed	Fixed

9. Bring up the **Graph Editor** again, with the *squashHandle1* selected, click **Curves->Post Infinity->Cycle**

10. Parent the *squashHandle1* to the mesh so that it follows it upward in the Y-axis.

11. Set the Second Squash handle deformer as follows:

select the mesh, **Create Deformers->non Linear->squash**

Once you have all the Attributes set up(Fig. 25), move the squash deformer to the bottom of the Hood mesh, so that the top end of the deformer is just overlapping the Margin (Edge of the Jellyfish) (Fig. 28).

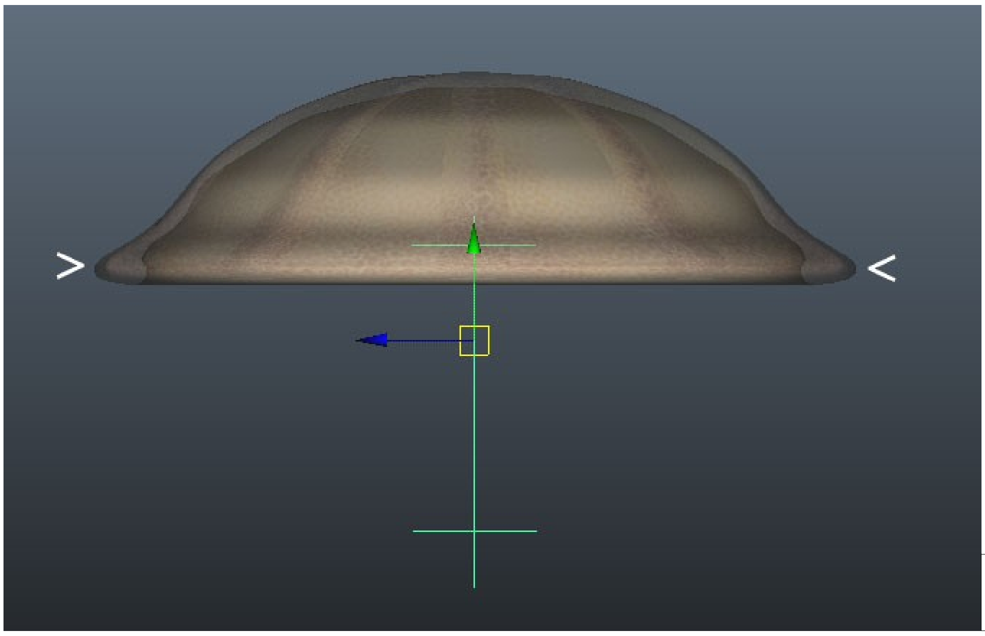
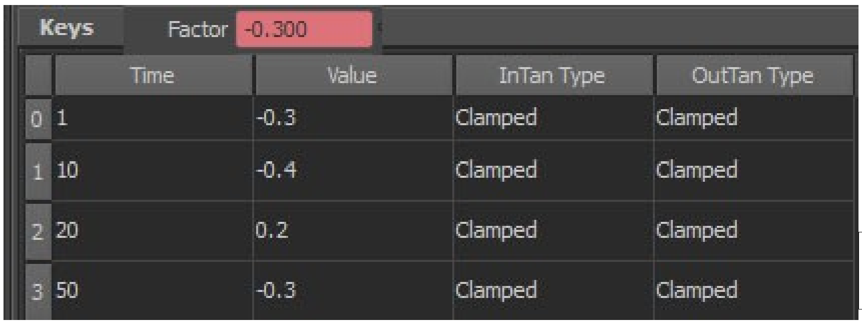


Fig. 28

12. And key-frame the *Factor*, using **RMB-click** over *Factor* **click->Set Key** on the frames in Fig. 29.



Keys Factor -0.300

	Time	Value	InTan Type	OutTan Type
0	1	-0.3	Clamped	Clamped
1	10	-0.4	Clamped	Clamped
2	20	0.2	Clamped	Clamped
3	50	-0.3	Clamped	Clamped

Fig. 29

13. Bring up the **Graph Editor** again, with the *squashHandle2* selected, click **Curves->Post Infinity->Cycle**

14. Parent the *squashHandle2* to the mesh so that it follows it upward in the Y-axis.

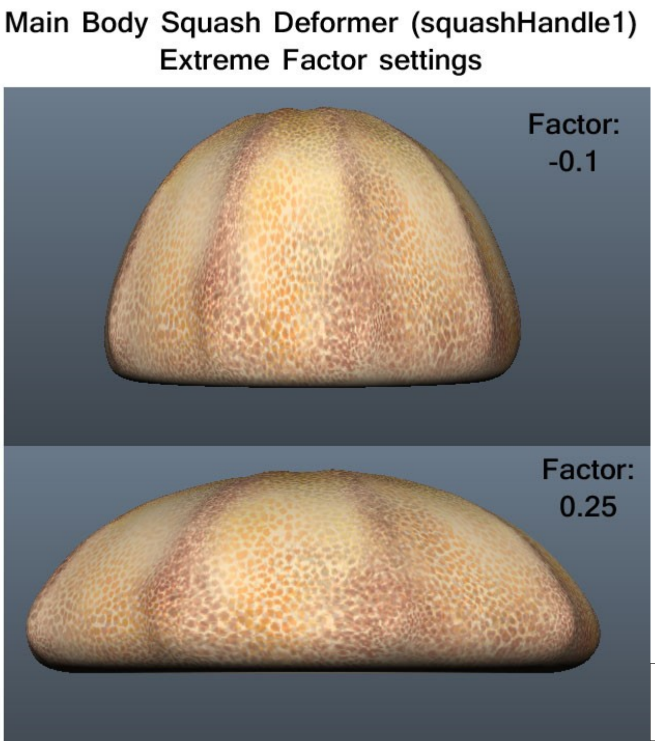
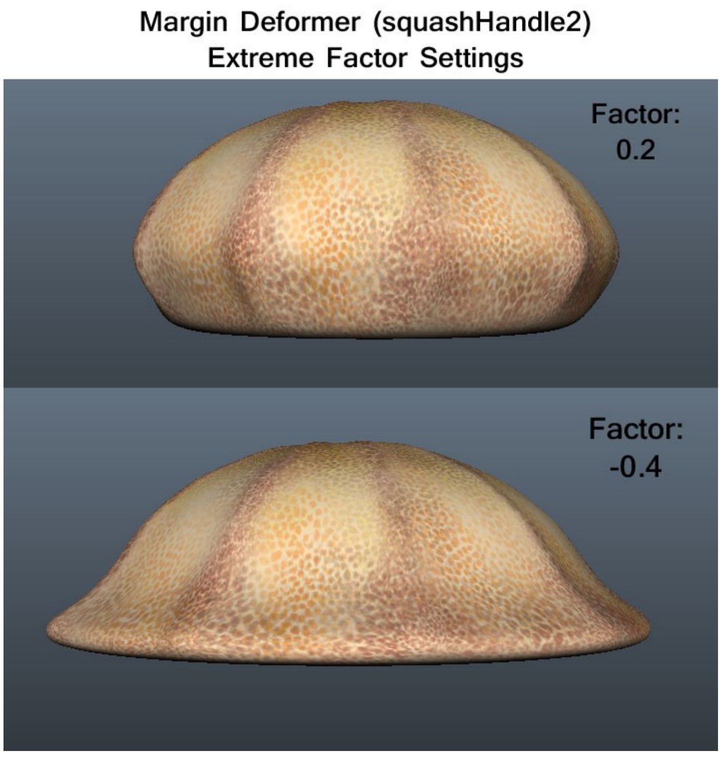


Fig. 30

Notice that the second deformer, *squashHandle2* (Fig. 31), affects only the very bottom Edge, or Margin, of the Jellyfish. This is the Effect you want, or the Jellyfish will not look right. If your squash deformer does not look like this when *factor* is set to the largest keyed positions, then you need to re-position the deformer's y-axis better.



(vii) Putting Everything Together

1. Inside our Mesh deformer scene, we need to create a handle for our tentacles to attach to. We are going to make a nurbsCircle controller.
2. Select **create NURBS primitives->circle** and scale it to roughly the same size as the edge Margin on our Jellyfish mesh (Fig. 32). Set the time scale to 0, select the nurbsCircle then **Modify->Freeze transformations**

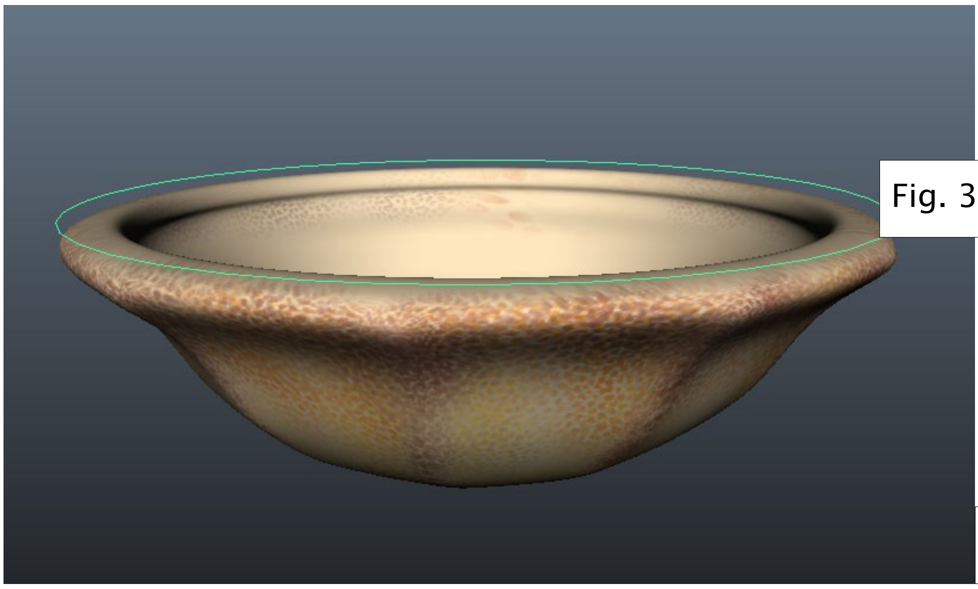


Fig. 31

Fig. 32

3. Now, in order to get the nurbsCircle to replicate the movement of the edge of the Jellyfish Margin, we have to manually key-frame , the scale factor every 10-frames up until frame 50, where our animation is recycled. The nurbs circle must stay as close as possible to the Margin, key-frame every 5 frame if you need to. When you have finished, the nurbsCircle should follow the edge of the Hood Margin perfectly. Select the nurbCircle then the mesh, press **P.** to parent. Now our nurbsCircle moves up in the Y at the same rate as the mesh. You may need to key frame the nurbsCircles local Y in some frames to clean-up where the deformers have extended the mesh in the Y-axis.

   Now we need to import out tentacles to attach to the nurbs circle controller.

-For Paint Effects Tentacles

4. Import the Paint Effect scene, with the two paint strokes and two polygon tentacles attached to a polygon plane.
5. Inside the poly plane shading attributes, change the transparency to 0, We don't want to see this plane in our render.

6. Now select the plane, and the Main Hood mesh, and hit **P** to parent the plane to the Jellyfish Hood.
7. Move the plane to somewhere in the middle of the Jellyfish Mesh (Fig. 33)

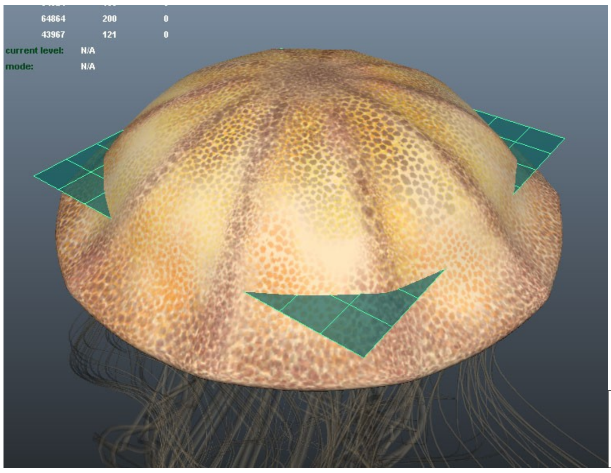
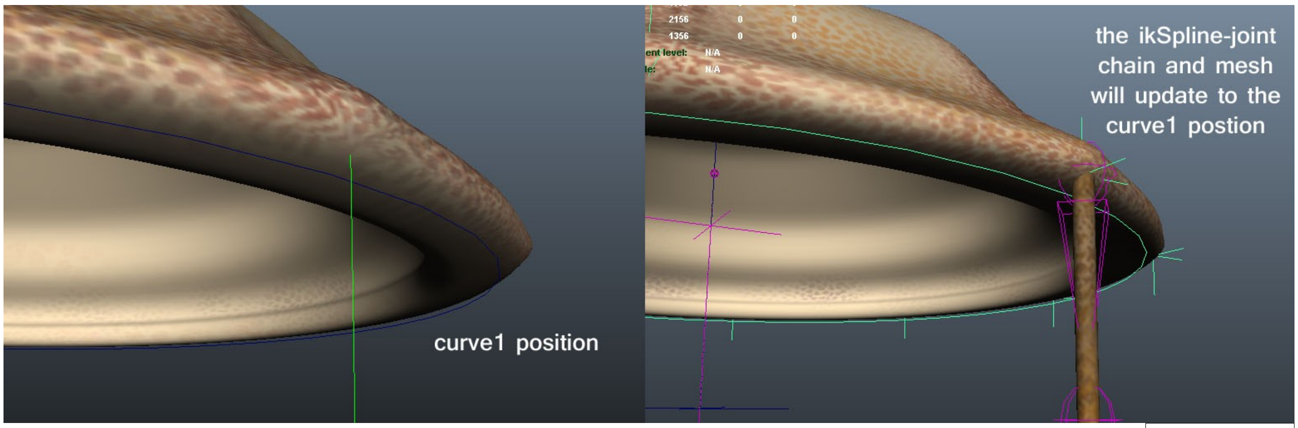


Fig. 33

8. Press play and the paint effects should now follow the path of the mesh and nurbs circle.

-For ikSpline Tentacles

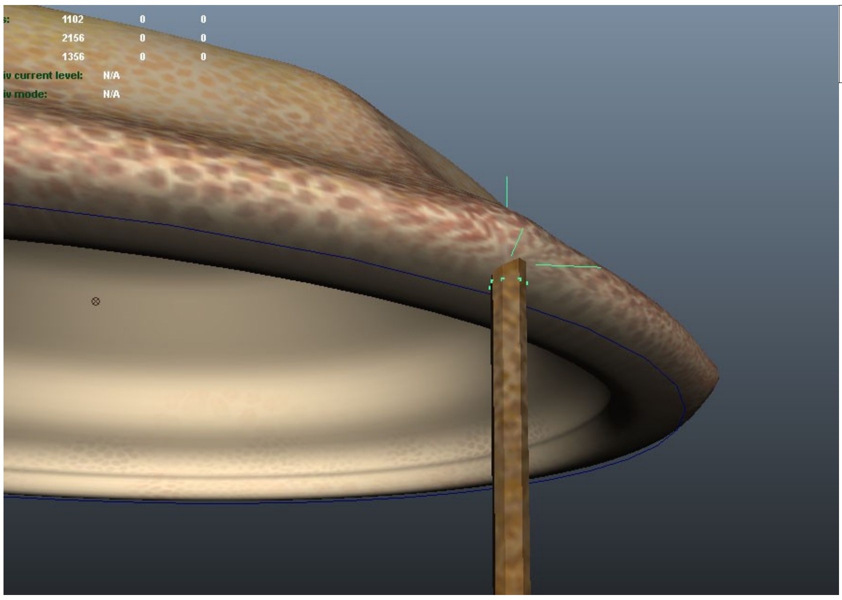
4. Import the ikSpline tentacle scene
5. Select *curve1* (the original curve that the hair follicle is attached to) and select the *nurbsCircle* controller then press **P** to parent.
6. Now move curve1 in its local XYZ-axis until its positioned at the edge of the Margin, where a tentacle would come out (Fig 34). The rest of the scene elements can be grouped together separately.


curve1 position
2156 0 0
1356 0 0
ent level: N/A
le: N/A
the ikSpline-joint chain and mesh will update to the curve1 postion
Fig. 34

7. Repeat this for as many ikSpline tentacles you wish to attach.

-For nCloth Tentacles

8. Import the nCloth tentacle scene
9. Select d*ynamicConstraint1* and select the *nurbsCircle* controller then press **P** to parent.
10. Now move *dynamicConstraint1* in its local XYZ-axis until its positioned at the edge of the Margin, where a tentacle would come out (Fig. 35).The rest of the scene elements can be grouped together separately.



```
s:               1102        0        0
                 2156        0        0
                 1356        0        0
iv current level:      N/A
iv mode:               N/A
```

Fig. 35

Expected Results at Frame 60

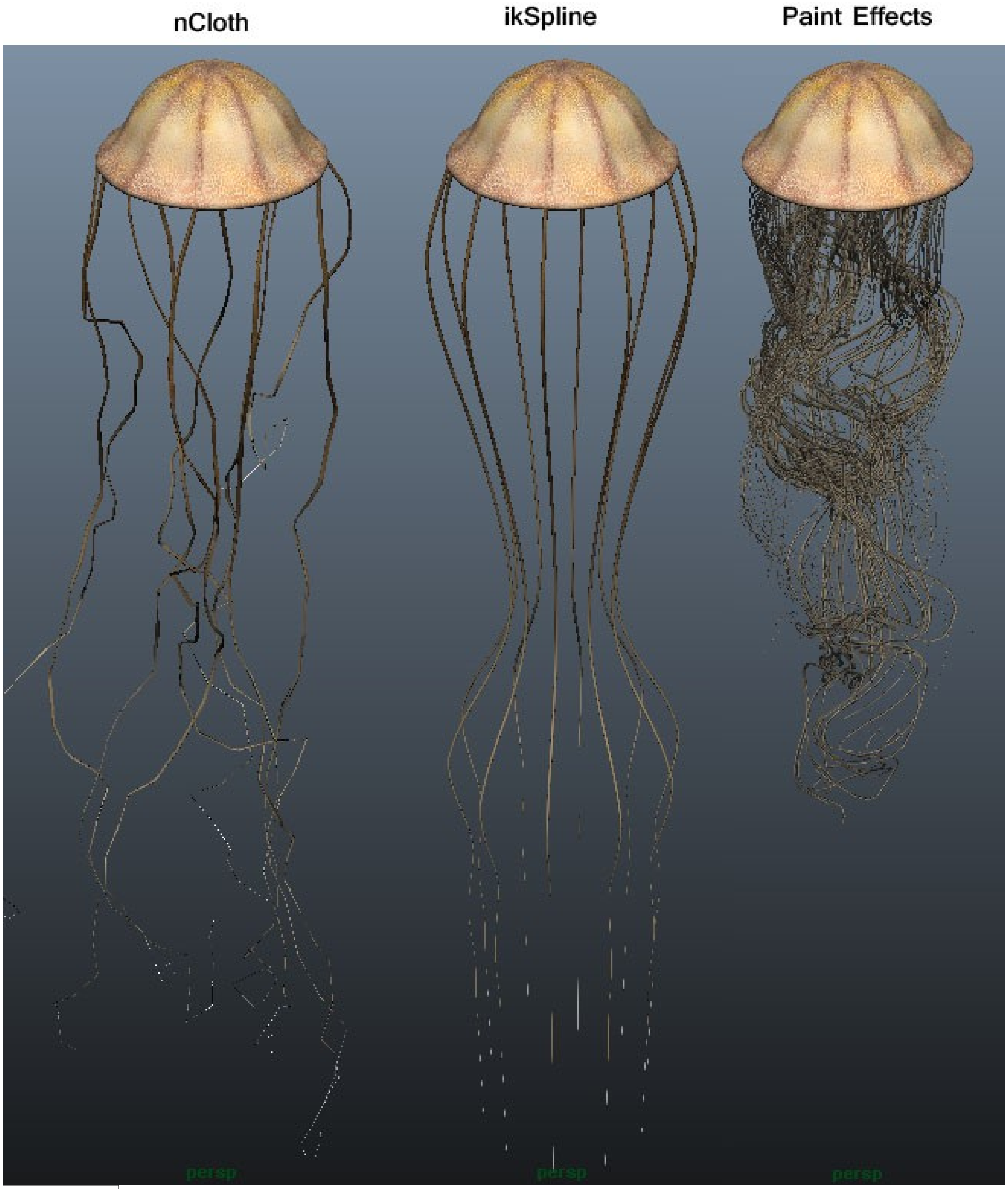


Fig. 36

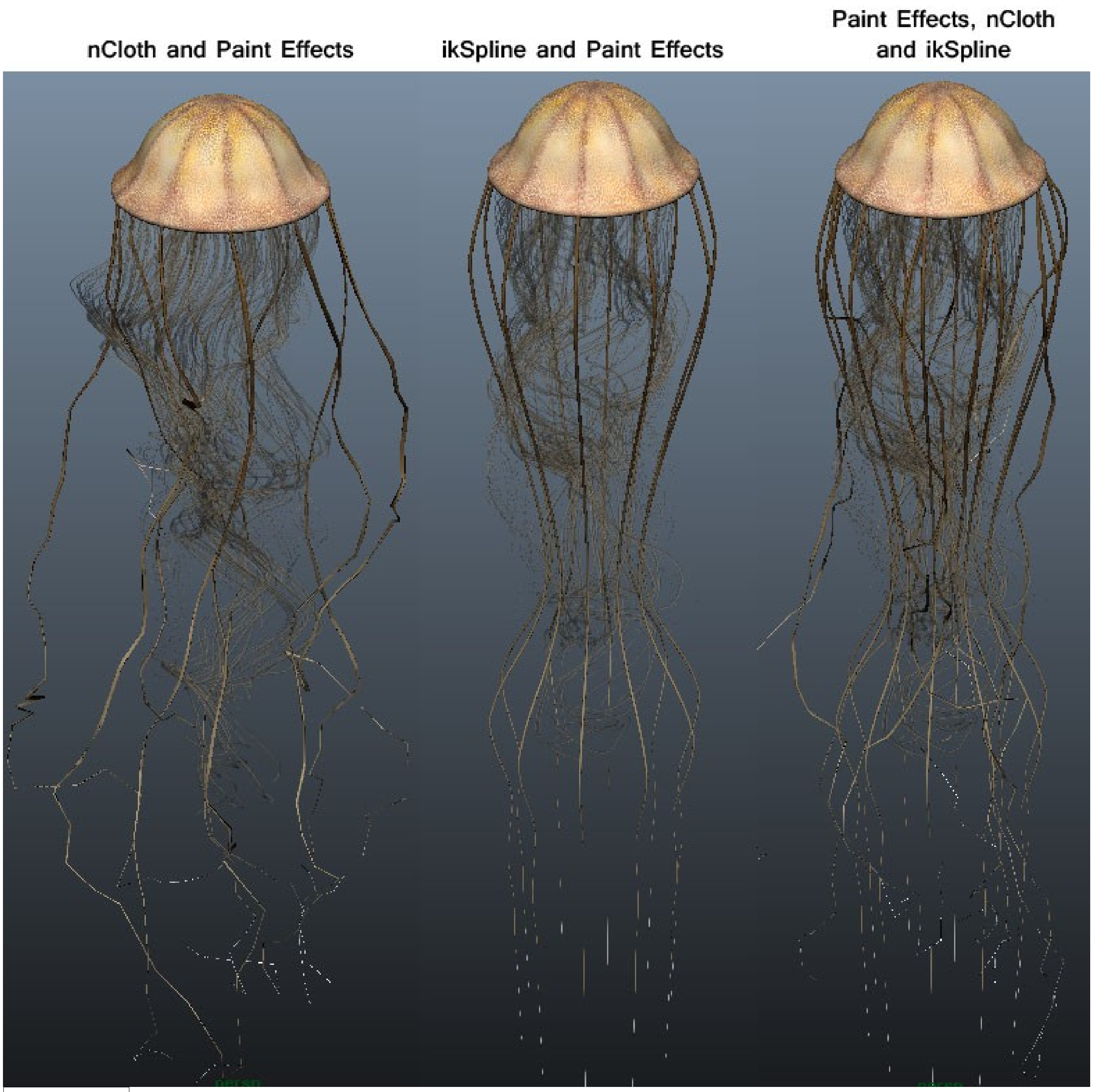

nCloth and Paint Effects
ikSpline and Paint Effects
Paint Effects, nCloth and ikSpline

Fig. 37

(viii) Potential Areas for Development

Sub Surface Scattering and Frilly-Oral Arms

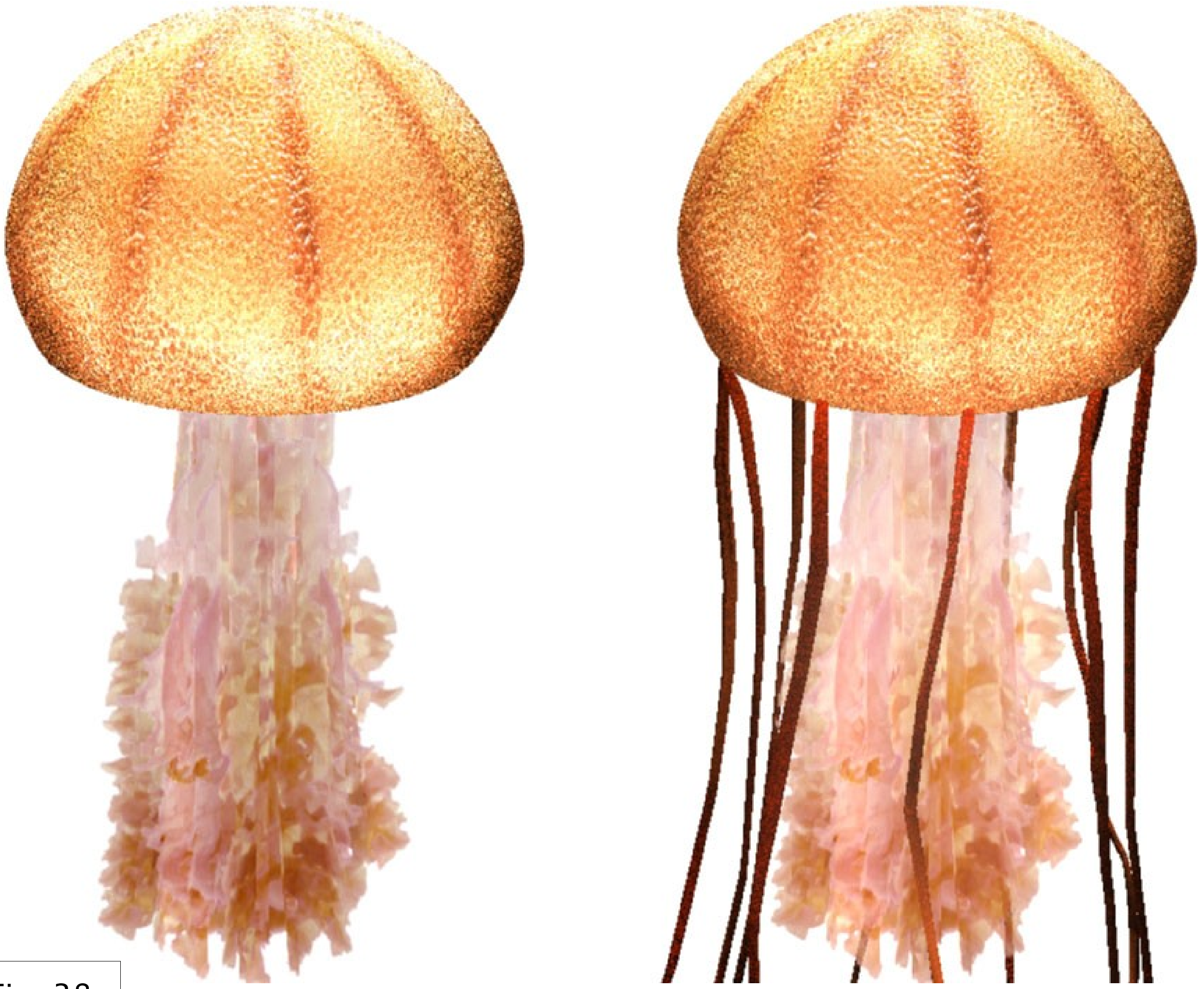


Fig. 38

To enhance the Look of the Jellyfish, a good area of Research would be in Sub-surface scattering. This creates the gentle 'glow' we see in a typical Pacific Sea Nettle (Fig. 40). I experimented with using a very basic Subsurface shader in Maya to create some nice effects (Fig. 38). The Frilly Oral Arms are simply a polygon plane nCloth attached to the body with a dynamic constraint. The diffuse was made in photoshop using an image of a lettuce leaf (Fig. 39), which I edited to the right colour, and then created an Alpha channel so that only the coloured areas show up in rendering.

Fig. 39

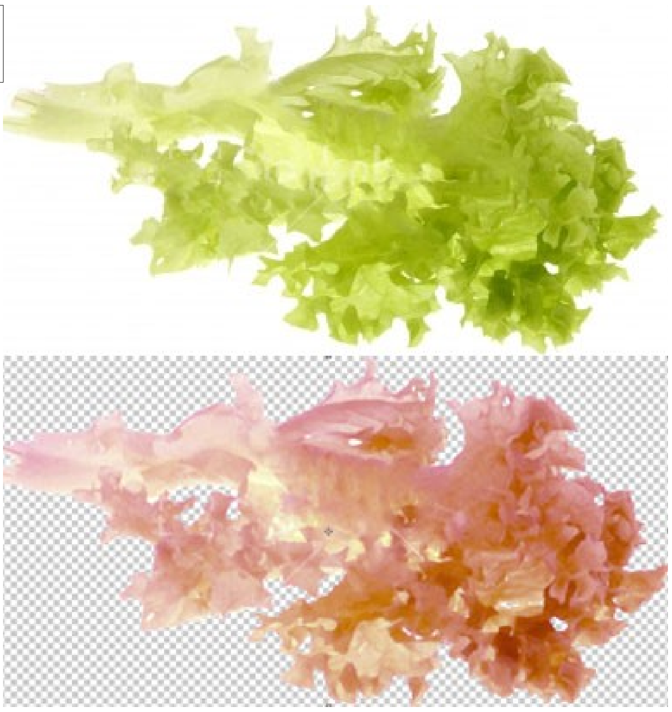
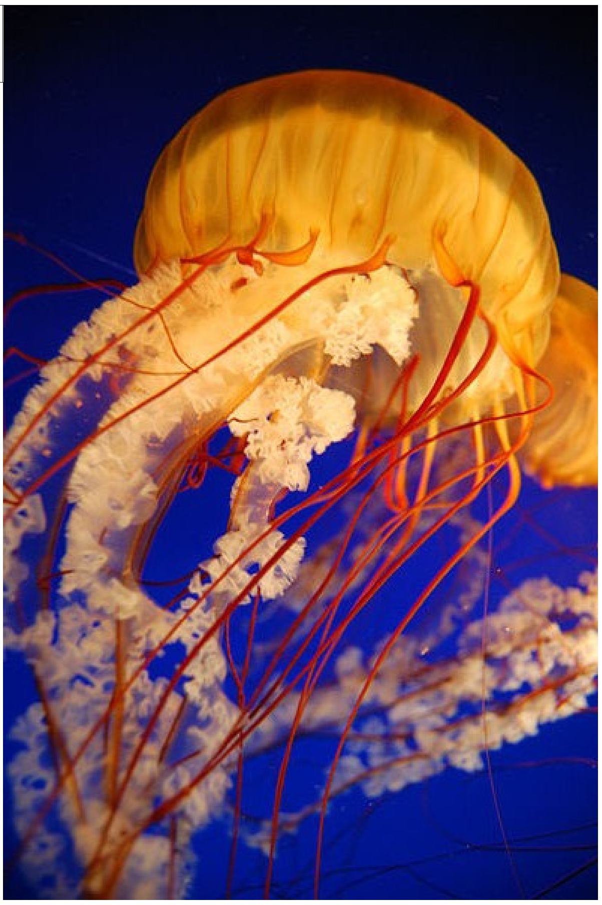


Fig. 40



Conclusion

The Aim of this project was focused towards extending my own knowledge into an area of Computer Graphics I had never ventured. I feel I have achieved this goal, and have provided a stable building-block for other students who wish to explore, research and develop a product in the same field. The research I made into existing Jellyfish in Computer Graphics merely scratches the surface of what is possible, as companies are very secretive about their methods, it can be hard to deduce how they created their Jellyfish. From studying footage of these existing CG Jellyfish, and studying footage of real-life Jellyfish, I was able to construct some solutions of my own, which I felt could achieve a similar result. I established that a Jellyfish is comprised of three main parts; The Hood, Oral Arms, and Tentacles. In order to make the tentacles and Oral Arms I decided to test out three different methods for making them; ikSpline, nCloth and Paint Effects. Through these tests, I was able to see the benefits and disadvantages of each type of tentacle produced (Fig. 36). The Paint Effects formed a very dense and twisted bundle which I thought would work well as Oral Arms, as they could be easily drawn and would fill a large section of the jellyfish with detail. The nCloth had a good result, the turbulence field helping to break up the monotony of identical solver attributes, they seem to work well as longer tentacles. The same could be said for the ikSpline, however, I felt that the look achieved was too symmetrical to be realistic, and so I used nCloth for my Jellyfish. Despite not wanting to use an ikSpline system myself, I felt the method was useful and other students could benefit from it so I kept it in my tutorial. I found that a combination of two or more of the tentacles really enriched the overall look of the Jellyfish (Fig. 37). I made it clear in my research that the most important thing for making my jellyfish was that the movement was convincing. I established a method for using two deformers to create the Jellyfish Hood movement. It was important to understand how they deform and move if I were to get a good result. My tutorial covers everything a student will need to model and rig a simple Jellyfish, it provides a good building block for anyone wishing to go on to develop hyper-realistic Jellyfish from it. If I had more room to develop this project further I would extend my tutorial to include methods for using Sub-surface scattering(Fig. 38) and creating more detailed oral arms (Fig. 39).

8) Bibliography and References

Books

Alias WaveFront, *Using Maya : Paint Effects,* (Silicon Graphics Limited, 2001) p22, p110-176

E. Keller Maya Visual Effects: The innovator's Guide (Wiley Publishing, 2007) p92-99

E. Keller *Mastering Maya 2011* (John Wiley & Sons; Pap/Dvdr edition, 2010) p296, p723-725, p758-763

Internet Resources

Walt Disney Picture's *Finding Nemo* (2003)
Source:http://vimeo.com/8402214

Hotpoint Aqualtis *Underwater World* (2008)
Source:http://www.youtube.com/watch?v=2qdHwrYafuQ

Ubisoft *Child of Eden Trailer* (2011)
Source: http://www.youtube.com/watch?v=xuYWLYjOa_0&ob=av3e

CG Society Publication, *The making of Finding Nemo* Source:
Source: http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/the_making_of_finding_nemo

Monterey Bay Aquarium, *Sea Nettle Exhibit*
Source:http://www.montereybayaquarium.org/animals/AnimalDetails.aspx?enc=LeWQvjcLBGScbfBrDNhDNA==
Video Footage Source: http://www.youtube.com/watch?v=rzNI1Yh0F5A

Animal Corner, *Jellyfish Anatomy*
Source:http://www.animalcorner.co.uk/marine/jellyfish/jellyfish_anatomy.html

Wikipedia, *Pacific sea nettle*
Source: http://en.wikipedia.org/wiki/Chrysaora_fuscescens