Workshop Series: Fur & Hair Shells

Benjamin Kenwright[1]*

Abstract
This practical focuses on real-time fur & hair effects using shells. The student needs to implement a real-time dynamic interactive fur simulator (e.g., add hair/fur effects to 3D models that bounces around). We present a technique for representing high detailed geometry using low-poly slicing. The complex geometry, i.e., fur and hair in this case, is created using layers, also called shells. Mapping textures onto these shells produces a visual representation of the high detailed model. These textured quads are rendered at relative offsets to the model's surface. The more slices give a more detailed visual representation of the model. This method enables us to create fur effects that run in real-time with high visual detail. In this practical, we show various enhancements to the basic shell method to generate more exotic, dynamic, and realistic fur effects (e.g., springs and forces).

Keywords
Fur, Hair, Shells, Soft Bodies, Interactive, Real-Time, Rigid Body Dyanmics, Rotation, Impulses, Collision Detection, Rigid Body Objects, Particles, Constraints, Classical Mechanics, Verlet

[1] *Workshop Series (www.xbdev.net) - Benjamin Kenwright*

Contents

Introduction

Hair & Fur Effects The topic of this practical is the implementation of a real-time 3D hair & fur simulation effect (i.e., interaction and control of a hair/fur effect in virtual environments). The user should be able to interact with the simulation while it is running (e.g., through the mouse or keyboard) to control the objects - such as, hair length and moving objects around so that they interact and visually display hair movement. The hair/fur simulator should be implemented using component programming principles (i.e., flexible set of library functions).

Tasks

- Visually display hair/fur effects on simple 3D models
- User input (e.g., mouse or keyboard) to control and move the objects around (i.e., ability to add forces to push objects)
- Drop different shapes into the scene and control the fur effect in real-time (e.g., length/colour/stiffness)
- Mix the shell technique with a mass-spring framework to create life-like dynamic movement (e.g., either spring-damper or verlet-based method)

Figure 1. Shell Fur & Hair Effects - Projecting multiple instances of the textured surface along the normal to create a computationally efficient and aesthetically pleasing visual fur/hair effect.

1. Overview

Principles and Concepts A robust and computationally efficient fur & hair effect is indispensable in interactive real-time virtual environments, such as games. A realistic effect that is computationally efficient and straightforward brings an otherwise static and in inflexible scene to life (i.e., objects would follow repetitive motions and wouldn't be interactive).

Dynamic Fur & Hair Shells We can expand the underpinning fur & hair shell principle to encapsulate basic dynamic behaviour by:

- Apply a position-based constraint system to the shell layers to create interactive fur movement (i.e., moves with the model or in response to forces, such as wind)
- Expand the fur & hair effect to create different effects (e.g., thick, think, and long) - e.g., hair that changes over

time or grows

Onions The shells are built up like layers on an onion to make up the model. The shells let us achieve high detail objects using a reduced poly model. This method demonstrates excellent real-time results. Starting with a fully transparent texture, noise pixels are scattered across the image, this image is then wrapped across shells which are along the normal of the surface. Varying the number of shells/layers shown in Figure 3 demonstrates varying realism using a 'fixed' image. Following on from a constant 'fixed' shell, we alter the shell so that the number of 'hair' pixels is reduced as you move further out towards the outer layers, shown in Figure 4. Fur can be combined with texture decal information to create textured fur effects, as shown in Figure 1.

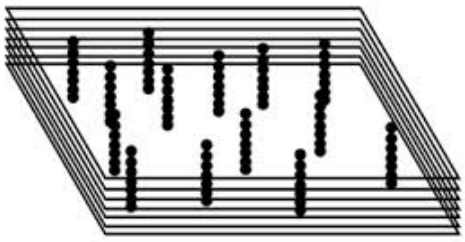


Figure 2. Shell Concept - Projecting multiple instances of the textured surface along the normal.

Figure 3. Shell Numbers - Varying the number of shells, offset, and distance apart can create a variety of different effects.

2. Implementation

For the practical exercise, we give you an uncomplicated OpenGL shader program that demonstrates Shelling. The program does:

1. Takes pre-created image and applies it to a basic shape (i.e., cube)
2. Alpha is enabled and alpha values less than 0 aren't written to the z-buffer
3. Culling is disable to draw the front and the back of the texture (possibly disable for the final effect)
4. Shells are a fixed distance with fixed texture mapped across them

Output Runs at real-time frame-rates and is ideal for games. Produce excellent visual effects. Can vary the fur types e.g. fine, thick, clumpy, and messy. Simple and intuitive, can be combined with textures to create textured fur. Different models for the fur pattern across the layers. Further work is to add fins and springs between the various layers to give an interactive fur effect. Create grass, tree effects. Alternative noise algorithms to generate shell images compare and investigate.

2.1 Physics

For the practical, you need to expand the basic shell implementation to make it a dynamic fur/hair effect. The tasks are in two parts, modifying the graphical implementation to include additional features (e.g., loading in models and texture/fur patters) and the physics-based part for synthesizing hair motion:

Graphics

1. Try different textures
2. Randomly generate a texture (i.e., lots of noise particles to create hair and apply it to the shells)
3. Vary the alpha (i.e., higher layer shells are with different alpha compared to lower ones)
4. Load in different models (obj or ply) and apply the technique
5. Texture the fur (i.e., one texture for the colour information and one texture for the alpha hair effect)
6. Experiment with hair length, thickness, colour, density, patterns

Physics/Movement

1. Oscillate the layers (e.g., use sin/cosine motion so they gradually oscillate and move)
2. Connect layers together with verlet distance constraints so the hair and fur bounces around as it moves
3. Apply the hair/fur effect to different scenes (e.g., a car or a characters head)

3. Summary

We have introduced an uncomplicated real-time fur & hair simulator for interactive dynamic scenes. Combining basic shells with springs and particles enables us to create an interactive effect suitable for real-time environments, such as games.

4. Exercises

This practical only gives a brief taste of fur & hair effects. As an exercise for the student to help enhance their understanding:

Intermediate

- Implement a scene with other shell based effects, such as, grass
- Experiment with creating different types of hair / fur effects (e.g., mixing shells with strands of hair)
- Explore inter-hair shadow effects

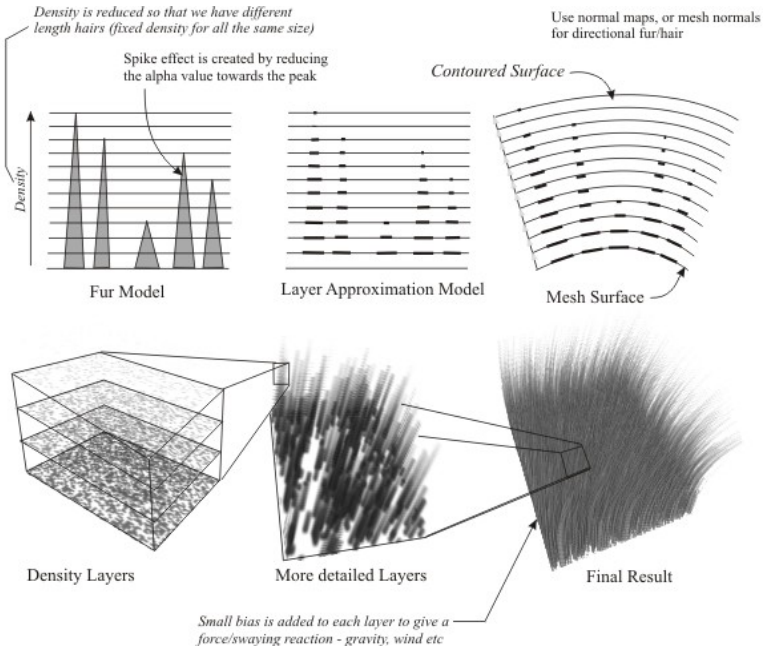
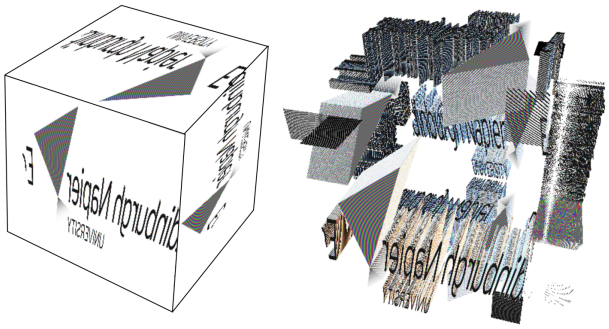


Figure 4. Details - Projecting multiple instances of the textured surface along the normal.



(a) (b)

Figure 5. OpenGL Shader Shell Implementation - Uncomplicated OpenGL implementation of shelling - i.e., a texture mapped onto multiple surface shells. The shape is a simple cube with shells of a fixed distance projected outwards at a fixed displacement based on the triangles normal. (a) A rendered cube, and (b) the same cube rendered with shells (i.e., 50 shells with fixed alpha and fixed distance apart).

Figure 6. Shell Concept - Projecting multiple instances of the textured surface along the normal.

Acknowledgements

We would like to thank all the students for taking time out of their busy schedules to provide valuable and constructive feedback to make this practical more concise, informative, and correct. However, we would be pleased to hear your views on the following:

- Is the practical clear to follow?
- Are the examples and tasks achievable?
- Do you understand the objects?
- Did we missed anything?
- Any surprises?

The practicals provide a basic introduction for getting started with physics-based animation effects. So if you can provide any advice, tips, or hints during from your own exploration of simulation development, that you think would be indispensable for a student's learning and understanding, please don't hesitate to contact us so that we can make amendments and incorporate them into future practicals.

Recommended Reading

NVIDIA White Paper: Fur (using Shells and Fins), http://developer.download.nvidia.com/SDK/10/direct3d/Source/Fur/doc/FurShellsAndFins.pdf, February 2007

Real-time Fur over Arbitrary Surfaces, Jerome Lengyel, Emil

Praun, Adam Finkelstein, Hugues Hoppe

Real-time Fur Rendering For Short Haired Creatures, Adam Lake, Kiefer Kuah

Fur Rendering and Dynamics Using Discrete Shells, JC Chong

Computer Animation: Algorithms & Techniques, Rick Parent, Publisher: Morgan Kaufmann, ISBN: 978-0124158429

Code Complete: A Practical Handbook of Software Construction, Steve McConnell, ISBN: 978-0735619678

Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, ISBN: 978-0132350884

Game Inverse Kinematics: A Practical Introduction (2nd Edition) Kenwright. ISBN: 979-8670628204

Kinematics and Dynamics Paperback. Kenwright. ISBN: 978-1539595496

Game Collision Detection: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1511964104

Game C++ Programming: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1516838165

Computational Game Dynamics: Principles and Practice (Paperback). Kenwright. ISBN: 978-1501018398

Game Physics: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1471033971

Game Animation Techniques: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1523210688

Figure 7. Shell Concept - Projecting multiple instances of the textured surface along the normal.