Universidad de Huelva

Tema 11

Skinning

Índice

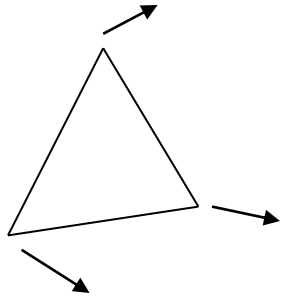
- En el renderizado las superficies se construyen a partir de primitivas simples como triángulos. También pueden utilizarse superficies lisas como NURBS o superficies de subdivisión, pero éstos a menudo acaban por convertirse en triángulos por un algoritmo de teselado (triangulación) automático antes del renderizado.

Iluminación

- Podemos calcular la interacción de la luz con las superficies para lograr sombreado realista.
- Para cálculos de iluminación, generalmente se requiere un posición sobre la superficie y la normal .
- Para imágenes de mas alta calidad, se debe realizar "iluminación global", donde la interacción completa de la luz es hallada dentro de un entorno para alcanzar efectos como sombras, reflejos y la luz difusa rebotada.

Sombreado de Gouraud & Phong

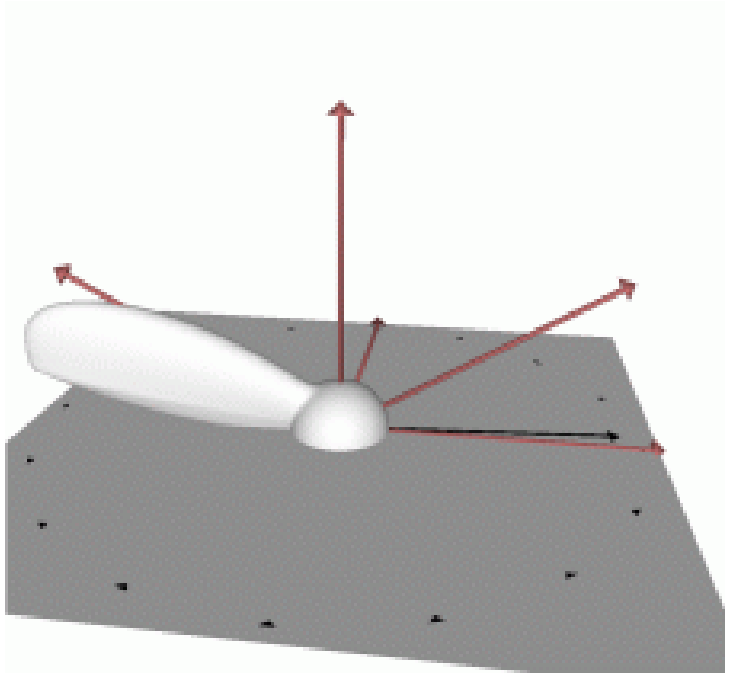
- Podemos usar triángulos para dar la apariencia de una superficie sombreada modificando las normales un poco.



- El Sombreado Gouraud es una técnica en la que se calcula la iluminación en cada vértice y se interpola el color resultante a través del triángulo.
- El Sombreado Phong es más costoso computacionalmente e interpola la normal a través del triángulo y se vuelve a calcular la iluminación para cada píxel.

Materiales

- Cuando un haz de luz entrante golpea una superficie, parte de la luz será absorbida, y algunas se dispersan en varias direcciones



Materiales

- En renderizado de alta calidad, se usa una función denominada BRDF (Función de distribución de reflectancia bidireccional) para representar a la dispersión de la luz en la superficie:

$$f_r(\theta_i, \varphi_i, \theta_r, \varphi_r, \lambda)$$

- La BRDF es una función de 5 dimensiones de la dirección de entrada de luz (2 dimensiones), la dirección de salida (2 dimensiones), y la longitud de onda.

Transparencia

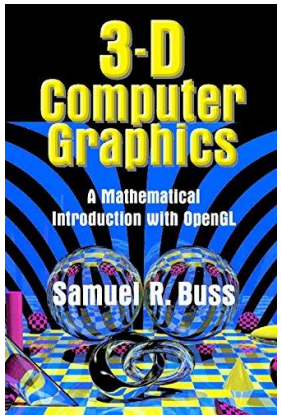
- La piel es un material translúcido. Si queremos hacer la piel realista, tenemos que dar cuenta de la dispersión de luz bajo la superficie.
- Podemos extender la BRDF a un BSSRDF añadiendo dos dimensiones más que representan la traducción en coordenadas de superficie. De esta manera, se puede dar cuenta de la luz que penetra en la superficie en un lugar determinado.

Texturas

- Podríamos desear 'asignar' varias propiedades a través de una superficie poligonal.
- Podemos hacerlo a través de la asignación de texturas, u otras técnicas de mapeos más generales.
- Generalmente, esto requerirá expresamente el almacenamiento de información de coordenadas de textura en los vértices.
- Para renderizado de alta calidad, podemos combinar varias asignaciones diferentes de diversas formas, cada una con su propio mapeo de coordenadas.

Texturas

- Funciones relacionadas incluyen mapeado de relieve (rugosidad), mapeo de desplazamiento, mapeo de iluminación …



• Más información: Samuel R. Buss, "3-D Computer Graphics. A Mathematical Introduction with OpenGL".

- Suma ponderada:

$$x' = \sum_{i=0} w_i x_i$$

- Promedio pesos:

$$\sum_{i=0} w_i = 1$$

- Promedio Convexo

$$0 \leq w_i \leq 1$$

Partes rígidas

- Robots y criaturas mecánicas pueden ser renderizados generalmente con partes rígidas y no requieren suavizado de la piel.
- Para renderizar partes rígidas, cada parte es transformada por su matriz articulación independientemente.
- En esta situación, cada vértice de la geometría del personaje es transformado exactamente por una matriz

$$\mathbf{v}' = \mathbf{W} \cdot \mathbf{v}$$

donde **v** está definido en el espacio local en la articulación.

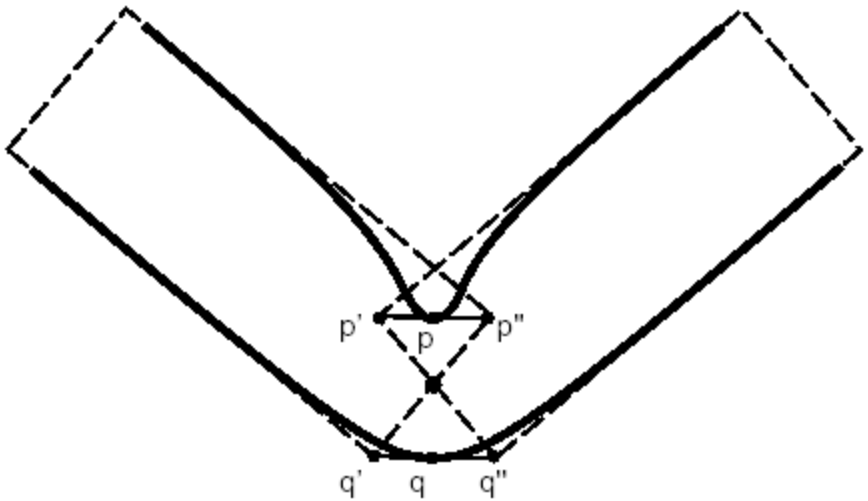
Simple skin

- Una mejora sencilla para personajes de calidad baja-media es unir rígidamente la piel al esqueleto. Esto significa que todos los vértices de la malla continua de la piel está unido a una articulación.
- En este método, como con partes rígidas, cada vértice está transformado exactamente una vez y por lo tanto debe tener un rendimiento similar al renderizado con partes rígidas.

$$\mathbf{v'} = \mathbf{W} \cdot \mathbf{v}$$

Suavizado de piel

- Con el algoritmo de suavizado de piel, un vértice puede conectar a más de una articulación con pesos ajustables, que controlan la cantidad con que cada articulación se ve afectada.



Suavizado de piel

- Los vértices raramente necesitan ser conectados a mas de tres articulaciones.
- Cada vértice se transforma un par de veces y los resultados se mezclan.
- El algoritmo de suavizado de piel tiene muchos otros nombres: blended skin, skeletal subspace deformation (SSD), multi-matrix skin, matrix palette skinning…

Suavizado de piel

- La posición del vértice deformado es una media ponderada:

$$\mathbf{v}' = w_1 (\mathbf{M_1} \cdot \mathbf{v}) + w_2 (\mathbf{M_2} \cdot \mathbf{v}) + \ldots + w_n (\mathbf{M_n} \cdot \mathbf{v})$$

$$\mathbf{v}' = \sum w_i (\mathbf{M_i} \cdot \mathbf{v})$$

donde

$$\sum w_i = 1$$

Binding matrices (aglutinantes)

- Con partes rígidas o de piel simple, **v** puede ser definido local a la articulación que la transforma.
- Con suavizado de piel, varias articulaciones transforman un vértice, pero no se puede definir local a todas ellas.
- En vez de esto, debemos primero transformarlo en local a la articulación que será transformada al espacio mundo.
- Para ello, se utiliza una matriz aglutinante B para cada conjunto que define dónde estaba la articulación cuando se une la piel y premultiplicar su inversa con la matriz del mundo:

$$M_i = W_i \cdot B_i^{-1}$$

Normales

- Para calcular el sombreado, también se necesita transformar las normales al espacio mundo
- Debido a que la normal es un vector de dirección, no queremos obtener la traslación desde la matriz, así que necesitamos solamente multiplicar la normal por la parte superior 3x3 de la matriz
- Para una normal ligada solo a una articulación:

$$\mathbf{n}' = \mathbf{W} \cdot \mathbf{n}$$

Normales

- Para el suavizado de piel, debemos mezclar la normal como con las posiciones, pero la normal debe ser renormalizada:

$$n' = \sum w_i \, (\mathbf{M_i} \cdot \mathbf{n}) \, / \, | \sum w_i \, (\mathbf{M_i} \cdot \mathbf{n}) |$$

Algorithm Overview

Skin::Update() (view independent processing)

- Computar la matriz de skinning para cada articulacion: $\mathbf{M}=\mathbf{W}\cdot\mathbf{B}^{-1}$ (se puede precomputar y almacenar $\mathbf{B}^{-1}$ en vez de $\mathbf{B}$)
- Recorrer los vértices y computar la posición y la normal mezclada

Skin::Draw() (view dependent processing)

- Poner la matriz estado a Identidad (world)
- Recorrer los triángulos y dibujar usando las posiciones y normales del espacio mundo

Rig Data Flow

- Entrada GDLs

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_N \end{bmatrix}$$

- Sistema de Rigging

  (skeleton, skin…)

Rig

- Salida: malla renderizable

  (vértices, normales…)

$$\mathbf{v}', \mathbf{n}'$$

Cinemática Directa en Esqueleto

- Cada articulación halla una matriz local basada en sus GDLs y cualquier otra constante necesaria (offset de la articulación…)

$$\mathbf{L} = \mathbf{L}_{\mathbf{joint}}(\phi_1, \phi_2, \ldots, \phi_N)$$

- Para encontrar la matriz mundo de la articulación, se halla el producto escalar de la matriz local con la matriz mundo del padre.

$$\mathbf{W} = \mathbf{W}_{\mathbf{parent}} \cdot \mathbf{L}$$

- Normalmente, se haría esto comenzando en la raíz y buscando primero en profundidad, así se estaría seguro que la matriz mundo del padre está disponible cuando fuera necesario.

- La posición de vértice deformado es una media ponderada sobre todas las articulaciones al que el vértice está unido:

$$\mathbf{v}' = \sum w_i \, (\mathbf{W_i} \cdot \mathbf{B_i}^{-1} \cdot \mathbf{v})$$

- **W** es una matriz mundo de articulación y **B** es una matriz de union de articulación que describe adonde fue su matriz mundo cuando fue unida al modelado de la piel (skin) (durante el tiempo de creación de la piel).
- Cada articulación transforma el vértice como si estuviera conectada rígidamente, y entonces esos resultados se mezclan basado en los pesos especificados por el usuario.

- Todos los pesos deben sumar 1:

$$\sum w_i = 1$$

- El combinado de las normales es esencialmente lo mismo, salvo que transformamos como vectores de dirección (x, y, z, 0) y luego se renormaliza los resultados.

$$\mathbf{n^*} = \sum w_i \, (\mathbf{W_i} \cdot \mathbf{B_i^{-1}} \cdot \mathbf{n})$$

$$\mathbf{n} = \mathbf{n^*} \, / \, |\mathbf{n^*}|$$

Ecuaciones para Skinning

- Skeleton

$$\mathbf{L} = \mathbf{L}_{\mathbf{joint}}(\phi_1, \phi_2, \ldots, \phi_N)$$

$$\mathbf{W} = \mathbf{W}_{\mathbf{parent}} \cdot \mathbf{L}$$

- Skinning

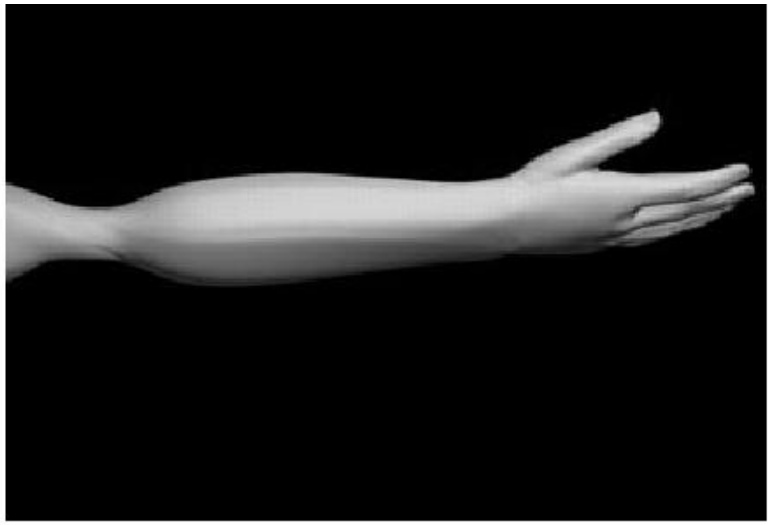
$$\mathbf{v}' = \sum w_i \, (\mathbf{W_i} \cdot \mathbf{B_i^{-1}} \cdot \mathbf{v})$$

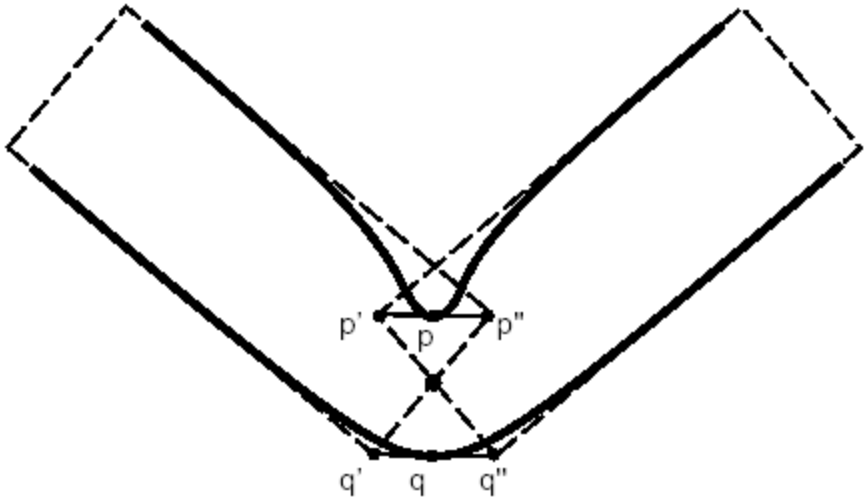
$$\mathbf{n}^* = \sum w_i \, (\mathbf{W_i} \cdot \mathbf{B_i^{-1}} \cdot \mathbf{n})$$

$$\mathbf{n} = \mathbf{n}^* / |\mathbf{n}^*|$$

Limitaciones del suavizado de piel

- El suavizado de piel es muy simple y rápido, pero su calidad es limitada.
- Los principales problemas son:
  - Las articulaciones tienden a colapsarse a medida que se doblan más.
  - Es muy difícil conseguir un control específico.
  - Poco intuitivo y difícil de editar.
- Aún así, está integrado en la mayoría de los paquetes de animación 3D y cuenta con el apoyo tanto en OpenGL y Direct3D.
- Si no hay otra cosa, es una buena base para construir posteriormente sistemas mas complejos.




p'
p
p"
q'
q
q"

Enlaces óseos

- Para ayudar con el problema de colapsado de la articulación, una opción es usar enlaces óseos.
- Los enlaces oseos son articulaciones extra insertadas en el esqueleto para ayudar con el skinning.
- Se pueden añadir de forma automática en función del alcance de la articulación del movimiento. Por ejemplo, podrían añadirse a fin de evitar cualquier articulación gire más de 60 grados.
- Este es un enfoque simple que se usa en algunos juegos en tiempo real, pero no soluciona otros problemas de suavizado de piel.

Interpolación de la forma

- Otra de las posibilidades es que el algoritmo de suavizado permita modelar los vértices con valores clave a lo largo del movimiento de la articulación.
- Para un codo, por ejemplo, se podría modelar recto, a continuación, modelar completamente doblada.
- Estas formas se interpolan localmente a los huesos antes de aplicar el skinning.

Músculos y otros efectos

- Se puede añadir efectos personalizados como protuberancias musculares como articulaciones adicionales.
- Por ejemplo, el biceps podría ser una traslación o escalado de articulación que se controla suavemente en la parte superior del brazo. Su movimiento podría estar relacionado con el movimiento de la rotación del codo.
- Con este enfoque, también se puede utilizar la piel para los músculos, las protuberancias de grasa, expresiones faciales, y formas simple de ropa.

Proceso de Rigging

- Para *riggear* un personaje *skinneado*, uno debe tener una malla de piel geométrica y un esqueleto.
- Generalmente, el skin es construido en una pose neutral, a menudo en una postura cómoda de pie.
- El esqueleto, sin embargo, podría ser construido en más de una pose cero donde se supone que los GDL de las articulaciones son 0, causando una postura muy rígida, y recta.
- Para unir la piel al esqueleto, el esqueleto primero debe ser planteada en una pose de unión (*binding pose*).
- Una vez se ha realizado, los vértices pueden ser asignados a las articulaciones con los pesos apropiados.

Skin binding

- La colocación de una piel a un esqueleto no es un problema trivial y usualmente requiere herramientas automatizadas combinadas con extensa configuración interactiva.
- Los algoritmos de Binding típicamente involucran enfoques heurísticos.
- Algunos enfoques generales:
  - Containment
  - Point-to-line mapping
  - Delay tetrahedralization

Mapeado por contenedores

- Con algoritmos de unión de contenedores, el usuario se aproxima manualmente el cuerpo con las primitivas de volumen para cada hueso (cilindros, elipsoides, esferas ...).
- Después, el algoritmo comprueba cada vértice con los volúmenes y lo adjunta a la mejor adaptación para el hueso.
- Algunos algoritmos de contención unen a un solo hueso y luego usa el suavizado en una segunda pasada. Otros unen a múltiples huesos directamente y establece los pesos de la piel.
- Para una versión más automatizada, los volúmenes pueden ser configurados inicialmente sobre la base de las longitudes de los huesos y las ubicaciones de los hijos.

Mapeado Punto-a-línea

- Una forma sencilla de conectar un piel es tratar a cada hueso como uno o más segmentos rectos y conectar cada vértice al segmento de línea más cercana.
- Un hueso está hecha de segmentos que conectan la articulación pivote a los pivotes de cada hijo.

Mapeado por tetrahedralización

- Esta es una técnica complicada de la geometría que construye una tetrahedralization del volumen con la piel.
- Los tetraedros conectan todos los vértices de la piel y los pivotes del esqueleto de una manera relativamente limpia 'Delaunay‘.
- La conectividad de la malla puede entonces ser analizada para determinar la mejor asignación para cada vértice.

Ajuste de la piel

- Suavizado de Malla: Una articulación primero se une de una manera bastante rígida (ya sea automática o manualmente) y después los pesas se suavizan algorítmicamente.
- Extracción: Identificación automática y la eliminación de los vértices adjuntos aislados.
- Pintura con Peso: Algunas herramientas de 3D permiten la visualización de los pesos como colores (0 ... 1 -> negro ... blanco). Estos pueden ser ajustados y 'pintadas' de forma interactiva.

Ajuste de la piel

- Manipulación directa: Estos algoritmos permiten al vértice que sea movido a una posición "correcta" después de que el hueso es doblado, y calculan automáticamente los pesos necesarios para llegar allí.

Hardware skinning

- El algoritmo de suavizado de piel es simple y lo suficientemente popular como para tener algún tipo de apoyo directo en en el renderizado 3D por hardware.
- En realidad, solo requiere operaciones de vector para multiplicar/sumar y así se puede implementar a bajo nivel.

Uso de la memoria

- Para cada vertice se necesita almacenar:
  - Datos de Rendering (posicion, normal, color, coords de texture, tangentes…)
  - Datos de Skinning (numero de asignaciones, indice de articulaciones, pesos…)
- Si limitamos el personaje a tener como maximos 256 huesos, se puede almacenar un índice de hueso como un byte.
- Si limitamos el peso a 256 valores distintos, podemos almacenar un peso como un byte (esto nos da una precisión de 0,004%, lo cual está bien).

Uso de la memoria

- Si suponemos que un vértice adjuntará a lo sumo 4 huesos, entonces podemos comprimir los datos para skining a

(1 +1) * 4 = 8 bytes por vértice (64 bits)

- De hecho, incluso podemos exprimir otros 8 bits de que, al no almacenar el peso final, ya que

$$w3 = 1 - w0 - w1 - w2$$

Ejemplo de Vertex Shader para suavizado de piel

```
#version 450
#extension GL_ARB_separate_shader_objects : enable

layout(binding = 0) uniform TransformInfo {
    mat4 MVP;
    mat4 ViewMatrix;
    mat4 ModelViewMatrix;
} Transform;

layout(binding = 1) readonly buffer JointInfo {
    mat4 matrices[];
} Joint;

…
```

Ejemplo de Vertex Shader para suavizado de piel

```
…

layout(location = 0) in vec3 inPosition;
layout(location = 1) in vec3 inNormal;
layout(location = 2) in vec2 inTexCoord;
layout(location = 3) in vec4 inJointWeights;
layout(location = 4) in ivec4 inJointIndices;

layout(location = 0) out vec3 Position;
layout(location = 1) out vec3 Normal;
layout(location = 2) out vec2 TexCoord;

…
```

Ejemplo de Vertex Shader para suavizado de piel

```
…

void main()
{
   mat4 skinMat =    inJointWeights.x * Joint.matrices[inJointIndices.x] +
                     inJointWeights.y * Joint.matrices[inJointIndices.y] +
                     inJointWeights.z * Joint.matrices[inJointIndices.z] +
                     inJointWeights.w * Joint.matrices[inJointIndices.w];

   vec4 n4 = Transform.ModelViewMatrix * skinMat * vec4(inNormal, 0.0);
   vec4 v4 = Transform.ModelViewMatrix * skinMat * vec4(inPosition,1.0);
   Normal = vec3(n4);
   Position = vec3(v4);
   TexCoord = inTexCoord;
   gl_Position = Transform.MVP * skinMat * vec4(inPosition, 1.0);
}
```