GRADIENTS IN SVG [SVG, 2011]
A color ramp is a function $c$

$$c : [0, 1] \rightarrow \text{sRGBA}$$

that maps the interval $[0, 1]$ to colors with transparency
A color ramp is a function $c$

$$c: [0, 1] \rightarrow \text{sRGBA}$$

that maps the interval $[0, 1]$ to colors with transparency

Defined by a list of $n$ stops

$$(t_i, c_i) \in [0, 1] \times \text{sRGBA}, \quad \text{with} \quad i \in \{1, \ldots, n\}, \quad t_i < t_{i+1}$$
A color ramp is a function $c$

$$c: [0, 1] \rightarrow \text{sRGBA}$$

that maps the interval $[0, 1]$ to colors with transparency

Defined by a list of $n$ stops

$$(t_i, c_i) \in [0, 1] \times \text{sRGBA}, \quad \text{with} \quad i \in \{1, \ldots, n\}, \quad t_i < t_{i+1}$$

c(t) is linear by parts

$$c(t) = \frac{(t_{i+1} - t) c_i + (t - t_i) c_{i+1}}{t_{i+1} - t_i}, \quad t_i \leq t < t_{i+1}$$
A wrapping function $s$

$$s: \mathbb{R} \rightarrow [0, 1]$$

maps a real number to the domain of the color ramp
Wrapping function (or spread method)

A wrapping function $s$

$$s: \mathbb{R} \rightarrow [0, 1]$$

maps a real number to the domain of the color ramp

E.g., *pad* (or *clamp*), *repeat* (or *wrap*), and *reflect* (or *mirror*)

$$pad(t) = \min(1, \max(0, t))$$
Wrapping function (or spread method)

A wrapping function $s$

$$s : \mathbb{R} \rightarrow [0, 1]$$

maps a real number to the domain of the color ramp

E.g., *pad* (or *clamp*), *repeat* (or *wrap*), and *reflect* (or *mirror*)

$$\text{pad}(t) = \min (1, \max (0, t))$$

$$\text{repeat}(t) = t - \lfloor t \rfloor$$
A wrapping function $s$

$$s: \mathbb{R} \rightarrow [0, 1]$$

maps a real number to the domain of the color ramp

E.g., pad (or clamp), repeat (or wrap), and reflect (or mirror)

$$\text{pad}(t) = \min(1, \max(0, t))$$
$$\text{repeat}(t) = t - \lfloor t \rfloor$$
$$\text{reflect}(t) = 2\left|\frac{1}{2} t - \lfloor \frac{1}{2} t + \frac{1}{2} \rfloor\right|$$
A linear gradient mapping is a function $\ell$

$$\ell : \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by 2 control points $p_1, p_2$
A linear gradient mapping is a function $\ell$

$$\ell : \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by 2 control points $p_1, p_2$
A linear gradient mapping is a function $\ell$

$$\ell: \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by 2 control points $p_1, p_2$

It computes the normalized projected length of $p - p_1$ into $p_2 - p_1$

$$\ell(p) = \frac{\langle p - p_1, p_2 - p_1 \rangle}{\langle p_2 - p_1, p_2 - p_1 \rangle}$$
A radial gradient mapping is a function $r$

$$r : \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by a center $c$, a radius $r$, and a focal point $f$
A radial gradient mapping is a function $r$

$$r : \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by a center $c$, a radius $r$, and a focal point $f$
A radial gradient mapping is a function $r$

$$r : \mathbb{R}^2 \rightarrow \mathbb{R}$$

parametrized by a center $c$, a radius $r$, and a focal point $f$

It computes the length ratio of from point $p$ to $f$ and $q$ to $f$

$$r(p) = \frac{\|p - f\|}{\|q - f\|}$$

where $q$ is the intersection between the ray from focal point $f$ to point $p$ and the circle centered at $c$ with radius $r$
Every shape includes a transformation $T_o$ that maps it from *object coordinates* (where the object is defined) to *scene coordinates* (where the object is placed on a scene).
Every shape includes a transformation $T_o$ that maps it from *object coordinates* (where the object is defined) to *scene coordinates* (where the object is placed on a scene).

Similarly, every paint includes a transformation $T_p$ that maps points from *paint coordinates* (where the color is computed) to *scene coordinates* (where the color is painted).
Every shape includes a transformation $T_o$ that maps it from object coordinates (where the object is defined) to scene coordinates (where the object is placed on a scene).

Similarly, every paint includes a transformation $T_p$ that maps points from paint coordinates (where the color is computed) to scene coordinates (where the color is painted).

If you want to apply a transformation $T$ to a shape and want its paint to move with it, simply compose

\[ T'_o = T \circ T_o \]
\[ T'_p = T \circ T_p \]
A linear gradient is a function

$$\mathbb{R}^2 \rightarrow \text{sRGBA}$$

formed by the composition of a paint transform $T_p$, a linear gradient mapping $\ell$, a wrapping function $s$, and a color ramp $c$

$$p \mapsto c\left(s(\ell(T_p^{-1} p))\right)$$
A linear gradient is a function

$$\mathbb{R}^2 \rightarrow \text{sRGBA}$$

formed by the composition of a paint transform $T_p$, a linear gradient mapping $\ell$, a wrapping function $s$, and a color ramp $c$

$$p \mapsto c\left(s(\ell(T_p^{-1} p))\right)$$

Show in Inkscape
EXAMPLES
A radial gradient is a function

\[ \mathbb{R}^2 \rightarrow \text{sRGBA} \]

formed by the composition of a paint transform \( T_p \), a radial gradient mapping \( r \), a wrapping function \( s \), and a color ramp \( c \)

\[ p \mapsto c \left( s \left( r \left( T_p^{-1} p \right) \right) \right) \]
A radial gradient is a function

\[ \mathbb{R}^2 \to \text{sRGBA} \]

formed by the composition of a paint transform \( T_p \), a radial gradient mapping \( r \), a wrapping function \( s \), and a color ramp \( c \)

\[ p \mapsto c\left(s\left(r\left(T_p^{-1} p)\right)\right) \]

Show in Inkscape
EXAMPLES
Evaluating gradient paints

How to efficiently evaluate a ramp

• Linear search, binary search, uniform sampling
Evaluating gradient paints

How to efficiently evaluate a ramp
  • Linear search, binary search, uniform sampling

How to efficiently evaluate linear and radial mappings?
  • How many parameters are really needed?
Gradients in PostScript and PDF
Type 1: Function-dictionary-based shading
  • Basically texture mapping
  • Show EPS file
  • Will discuss in following classes
SHADING TYPES

Type 1: Function-dictionary-based shading
  - Basically texture mapping
  - Show EPS file
  - Will discuss in following classes

Type 2: Axial shading
  - Same as linear gradient
  - Show EPS file
Type 3: Radial shading
  • *Not* the same radial gradient
Type 3: Radial shading

- *Not* the same radial gradient
- Define $\gamma(p, r)$ to be the circle centered at $p$ with radius $r$
Type 3: Radial shading

- *Not* the same radial gradient
- Define $\gamma(p, r)$ to be the circle centered at $p$ with radius $r$
- Inputs are centers and radii for 2 circles $(p_1, r_1), (p_2, r_2)$
Type 3: Radial shading

\begin{itemize}
  \item \textit{Not} the same radial gradient
  \item Define \( \gamma(p, r) \) to be the circle centered at \( p \) with radius \( r \)
  \item Inputs are centers and radii for 2 circles \((p_1, r_1), (p_2, r_2)\)
  \item Maps the “interpolated” circle to the color from a ramp \( c \)
  \[ \gamma((1 - t)(p_1, r_1) + t(p_2, r_2)) \mapsto c(t) \]
\end{itemize}
Type 3: Radial shading

- Not the same radial gradient
- Define $\gamma(p, r)$ to be the circle centered at $p$ with radius $r$
- Inputs are centers and radii for 2 circles $(p_1, r_1), (p_2, r_2)$
- Maps the “interpolated” circle to the color from a ramp $c$

$$\gamma\left((1 - t)(p_1, r_1) + t(p_2, r_2)\right) \mapsto c(t)$$

- Show EPS file
Type 4: Free-form Gouraud-shaded triangle mesh
- Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
Type 4: Free-form Gouraud-shaded triangle mesh

- Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
- Maps convex combination of points to same combination of colors

- Triangles can be independent, strips, or fans

- Show EPS file and PDF file

Type 5: Lattice-form Gouraud-shaded triangle mesh

- Same, but for a “regular” grid of triangles
Type 4: Free-form Gouraud-shaded triangle mesh

- Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
- Maps convex combination of points to same combination of colors
- I.e., given \(0 < s, t < 1\), Gouraud maps

\[ p(s, t) \mapsto c(s, t) \]

with

\[ p(s, t) = sp_1 + tp_2 + (1 - s - t)p_3 \]
\[ c(s, t) = sc_1 + tc_2 + (1 - s - t)c_3 \]
Shading types

Type 4: Free-form Gouraud-shaded triangle mesh
• Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
• Maps convex combination of points to same combination of colors
• I.e., given \(0 < s, t < 1\), Gouraud maps

\[
p(s, t) \mapsto c(s, t)
\]

with

\[
p(s, t) = sp_1 + tp_2 + (1 - s - t)p_3
\]
\[
c(s, t) = sc_1 + tc_2 + (1 - s - t)c_3
\]

• Triangles can be independent, strips, or fans
Type 4: Free-form Gouraud-shaded triangle mesh
• Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
• Maps convex combination of points to same combination of colors
• I.e., given \(0 < s, t < 1\), Gouraud maps
\[
p(s, t) \mapsto c(s, t)
\]
with
\[
p(s, t) = sp_1 + tp_2 + (1 - s - t)p_3
\]
\[
c(s, t) = sc_1 + tc_2 + (1 - s - t)c_3
\]
• Triangles can be independent, strips, or fans
• Show EPS file and PDF file
Shading types

Type 4: Free-form Gouraud-shaded triangle mesh

- Inputs are 3 vertices with colors \((p_1, c_1), (p_2, c_2), (p_3, c_3)\)
- Maps convex combination of points to same combination of colors
- I.e., given \(0 < s, t < 1\), Gouraud maps
  \[
  p(s, t) \mapsto c(s, t)
  \]
  with
  \[
  p(s, t) = s p_1 + t p_2 + (1 - s - t) p_3 \\
  c(s, t) = s c_1 + t c_2 + (1 - s - t) c_3
  \]
- Triangles can be independent, *strips*, or *fans*
- Show EPS file and PDF file

Type 5: Lattice-form Gouraud-shaded triangle mesh

- Same, but for a “regular” grid of triangles
EXAMPLES
Examples
Type 6: Coons patch mesh

- Each patch is defined by 4 connected cubic Bézier segments

\[ h_0(s), \ h_1(s), \ v_0(t), \ \text{and} \ v_1(t) \]
Type 6: Coons patch mesh

- Each patch is defined by 4 connected cubic Bézier segments $h_0(s)$, $h_1(s)$, $v_0(t)$, and $v_1(t)$

- Curves are setup to share endpoints like such

  \[
  v_{00} = v_0(0) = h_0(0) \quad v_{01} = v_0(1) = h_1(0) \\
  v_{10} = v_1(0) = h_0(1) \quad v_{11} = v_1(1) = h_1(1)
  \]
Type 6: Coons patch mesh

- Each patch is defined by 4 connected cubic Bézier segments
  \[ h_0(s), \ h_1(s), \ v_0(t), \ \text{and} \ v_1(t) \]
- Curves are setup to share endpoints like such
  \[ v_{00} = v_0(0) = h_0(0) \quad v_{01} = v_0(1) = h_1(0) \]
  \[ v_{10} = v_1(0) = h_0(1) \quad v_{11} = v_1(1) = h_1(1) \]
- Define \( h : [0, 1]^2 \to \mathbb{R}^2 \) to interpolate between curves \( h_0, h_1 \)
  \[ h(s, t) = (1 - t) h_0(s) + t h_1(s) \]
Type 6: Coons patch mesh

- Each patch is defined by 4 connected cubic Bézier segments

\[ h_0(s), \ h_1(s), \ v_0(t), \ \text{and} \ v_1(t) \]

- Curves are setup to share endpoints like such

\[
\begin{align*}
  v_{00} &= v_0(0) = h_0(0) & v_{01} &= v_0(1) = h_1(0) \\
  v_{10} &= v_1(0) = h_0(1) & v_{11} &= v_1(1) = h_1(1)
\end{align*}
\]

- Define \( h: [0, 1]^2 \to \mathbb{R}^2 \) to interpolate between curves \( h_0, h_1 \)

\[
h(s, t) = (1 - t) \ h_0(s) + \ t \ h_1(s)
\]

- Define \( v: [0, 1]^2 \to \mathbb{R}^2 \) to interpolate between \( v_0, v_1 \)

\[
v(s, t) = (1 - s) \ v_0(t) + \ s \ v_1(t)
\]
Type 6: Coons patch mesh

- Each patch is defined by 4 connected cubic Bézier segments
  \[ h_0(s), \quad h_1(s), \quad v_0(t), \quad \text{and} \quad v_1(t) \]

- Curves are setup to share endpoints like such
  \[
  v_{00} = v_0(0) = h_0(0) \quad \quad \quad \quad \quad \quad \quad v_{01} = v_0(1) = h_1(0) \\
  v_{10} = v_1(0) = h_0(1) \quad \quad \quad \quad \quad \quad \quad v_{11} = v_1(1) = h_1(1)
  \]

- Define \( h : [0, 1]^2 \rightarrow \mathbb{R}^2 \) to interpolate between curves \( h_0, h_1 \)
  \[ h(s, t) = (1 - t) h_0(s) + t h_1(s) \]

- Define \( v : [0, 1]^2 \rightarrow \mathbb{R}^2 \) to interpolate between \( v_0, v_1 \)
  \[ v(s, t) = (1 - s) v_0(t) + s v_1(t) \]

- Note that \( v(s, t) \) and \( h(s, t) \) interpolate all shared vertices
Type 6: Coons patch mesh (continued)

- Define bilinear map $m : V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2$

$$m^{a,b}_{c,d}(s, t) = (1 - s)(1 - t) a + (1 - s) t b + s (1 - t) c + s t d$$
Type 6: Coons patch mesh (continued)

- Define bilinear map $m : V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2$
  
  $$m_{a,b}^{c,d}(s, t) = (1 - s)(1 - t) a + (1 - s) t b + s (1 - t) c + s t d$$

- The bilinear map $m_{v_00, v_01}^{v_{10}, v_{11}}(s, t)$ also interpolates the shared vertices
Type 6: Coons patch mesh (continued)

• Define bilinear map $m : V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2$

\[
m^{a,b}_{c,d}(s, t) = (1 - s)(1 - t) a + (1 - s) t b + s (1 - t) c + s t d
\]

• The bilinear map $m^{V_{00}, V_{01}}_{V_{10}, V_{11}}(s, t)$ also interpolates the shared vertices

• Therefore, so does

\[
p(s, t) = v(s, t) + h(s, t) - m^{V_{00}, V_{01}}_{V_{10}, V_{11}}(s, t)
\]
Type 6: Coons patch mesh (continued)

- Define bilinear map \( m: V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2 \)
  \[
  m_{c,d}^{a,b}(s, t) = (1 - s)(1 - t) a + (1 - s) t b + s (1 - t) c + s t d
  \]
- The bilinear map \( m_{v_{10},v_{11}}^{v_{00},v_{01}}(s, t) \) also interpolates the shared vertices
- Therefore, so does
  \[
  p(s, t) = v(s, t) + h(s, t) - m_{v_{10},v_{11}}^{v_{00},v_{01}}(s, t)
  \]
- Given colors \( c_{00}, c_{01}, c_{10}, \) and \( c_{11}, \) the patch maps
  \[
  p(s, t) \mapsto m_{c_{10},c_{11}}^{c_{00},c_{01}}(s, t)
  \]
Shading types

Type 6: Coons patch mesh (continued)

- Define bilinear map \( m : V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2 \)
  \[
m_{a,b}^{c,d}(s, t) = (1 - s)(1 - t) a + (1 - s) t b + s (1 - t) c + s t d
  \]
- The bilinear map \( m_{V_{00},V_{01}}^{V_{10},V_{11}}(s, t) \) also interpolates the shared vertices
- Therefore, so does
  \[
p(s, t) = v(s, t) + h(s, t) - m_{V_{10},V_{11}}^{V_{00},V_{01}}(s, t)
  \]
- Given colors \( c_{00}, c_{01}, c_{10}, \) and \( c_{11}, \) the patch maps
  \[
p(s, t) \mapsto m_{c_{10},c_{11}}^{c_{00},c_{01}}(s, t)
  \]
- Patches can be defined independently or connected by strips
Type 6: Coons patch mesh (continued)

- Define bilinear map $m : V^4 \times [0, 1]^2 \rightarrow \mathbb{R}^2$
  \[ m_{a,b}^{c,d}(s,t) = (1 - s)(1 - t)a + (1 - s)t b + s(1 - t)c + std \]
- The bilinear map $m_{V_{00},V_{01}}^{V_{10},V_{11}}(s,t)$ also interpolates the shared vertices
- Therefore, so does
  \[ p(s,t) = v(s,t) + h(s,t) - m_{V_{10},V_{11}}^{V_{00},V_{01}}(s,t) \]
- Given colors $c_{00}$, $c_{01}$, $c_{10}$, and $c_{11}$, the patch maps
  \[ p(s,t) \mapsto m_{c_{10},c_{11}}^{c_{00},c_{01}}(s,t) \]
- Patches can be defined independently or connected by strips
- Show EPS file
Type 7: Tensor-product patch mesh

- This is just a generalization of Bézier curves to patches
Shading types

Type 7: Tensor-product patch mesh

- This is just a generalization of Bézier curves to patches
- Given control points \( p_{i,j} \), for \( i, j \in \{0, 1, 2, 3\} \), the tensor product is

\[
p(s, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} b_{i,3}(s) b_{j,3}(t)
\]

where \( b_{i,3}, b_{j,3} \) are the cubic Bernstein polynomials
Type 7: Tensor-product patch mesh

- This is just a generalization of Bézier curves to patches
- Given control points $p_{i,j}$, for $i, j \in \{0, 1, 2, 3\}$, the tensor product is

$$p(s, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} b_{i,3}(s) b_{j,3}(t)$$

where $b_{i,3}, b_{j,3}$ are the cubic Bernstein polynomials
- Given colors $c_{00}, c_{01}, c_{10}, c_{11}$, the patch maps

$$p(s, t) \mapsto m_{c_{00}, c_{01}}^{c_{10}, c_{11}}(s, t)$$
Type 7: Tensor-product patch mesh

- This is just a generalization of Bézier curves to patches
- Given control points \( p_{i,j} \), for \( i, j \in \{0, 1, 2, 3\} \), the tensor product is

\[
p(s, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} b_{i,3}(s) b_{j,3}(t)
\]

where \( b_{i,3}, b_{j,3} \) are the cubic Bernstein polynomials
- Given colors \( c_{00}, c_{01}, c_{10}, c_{11} \), the patch maps

\[
p(s, t) \mapsto m_{c_{00}, c_{01}}^{c_{10}, c_{11}}(s, t)
\]

- Patches can be defined independently or connected by strips
Shading types

Type 7: Tensor-product patch mesh

- This is just a generalization of Bézier curves to patches
- Given control points $p_{i,j}$, for $i, j \in \{0, 1, 2, 3\}$, the tensor product is

$$p(s, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} b_{i,3}(s) b_{j,3}(t)$$

where $b_{i,3}, b_{j,3}$ are the cubic Bernstein polynomials

- Given colors $c_{00}, c_{01}, c_{10}, c_{11}$, the patch maps

$$p(s, t) \mapsto m_{c_{00},c_{01}}^{c_{10},c_{11}}(s, t)$$

- Patches can be defined independently or connected by strips
- (Coons patch is a special case of tensor-product patch)
Shading types

Type 7: Tensor-product patch mesh

• This is just a generalization of Bézier curves to patches
• Given control points \( p_{i,j} \), for \( i, j \in \{0, 1, 2, 3\} \), the tensor product is

\[
p(s, t) = \sum_{i=0}^{3} \sum_{j=0}^{3} p_{i,j} b_{i,3}(s) b_{j,3}(t)
\]

where \( b_{i,3}, b_{j,3} \) are the cubic Bernstein polynomials

• Given colors \( c_{00}, c_{01}, c_{10}, c_{11} \), the patch maps

\[
p(s, t) \mapsto m_{c_{00}, c_{01}, c_{10}, c_{11}}(s, t)
\]

• Patches can be defined independently or connected by strips
• (Coons patch is a special case of tensor-product patch)
• Show EPS file
EXAMPLES
References

