# Lighting Effects for Mobile Games

"Shadows for Sprites" and "Fog from Above"

Jeppe Revall Frisvad, Niels Jørgen Christensen, and Peter Falster

Presenter: Jeppe Revall Frisvad



## In the Old Days

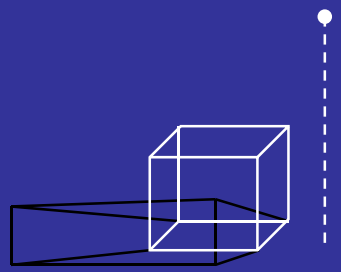
- Jim Blinn





## In the Old Days

- Jim Blinn
  - Planar Projection Shadows (IEEE Computer Graphics and Applications 1988)







## In the Old Days

- Jim Blinn
  - Planar Projection Shadows (IEEE Computer Graphics and Applications 1988)
  - Clouds and Dusty Surfaces (SIGGRAPH 1982)

## In the Old Days

- Jim Blinn
  - Planar Projection Shadows (IEEE Computer Graphics and Applications 1988)
  - Clouds and Dusty Surfaces (SIGGRAPH 1982)

- Ideas which are as relevant as ever, why is that?





## Mobile Devices

- We want small handheld devices ... with games and neat graphics
- Small handheld device means
  - Limited battery life
  - Limited processing power
  - Limited memory and storage capacity





## Common Mobile Games Characteristics

- Two-dimensional
- Tile-based
- Viewed from above
- Recently: 3D ...

Darkest Fear (Rovio)

Splinter Cell (Alltel)





## Mobile Games Characteristics

- Two-dimensional
- Tile-based
- Viewed from above

Darkest Fear (Rovio)


Duke Nukem (Alltel)

- Recently: 3D ... but we quickly lose track of things

## Tile-Based 2D Games Viewed from Above


- This type of game will always suit a small screen





## 3D Lighting Effects for 2D tile-based games

- Speaking in favour of
  - Everything is rendered in a plane
  - Sprites are few and reappearing
  - Alpha-blending is available
- Speaking against
  - No 3D information
  - Little memory available
  - Calculations have limited support

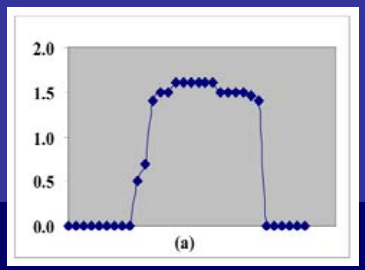




## Shadows for sprites

- Examples of sprites:

- Height information with little memory consumption:

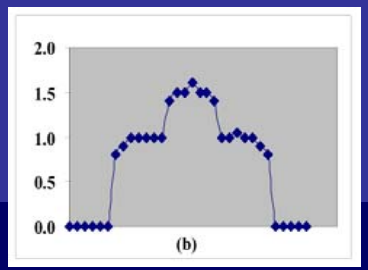





## Shadows for sprites

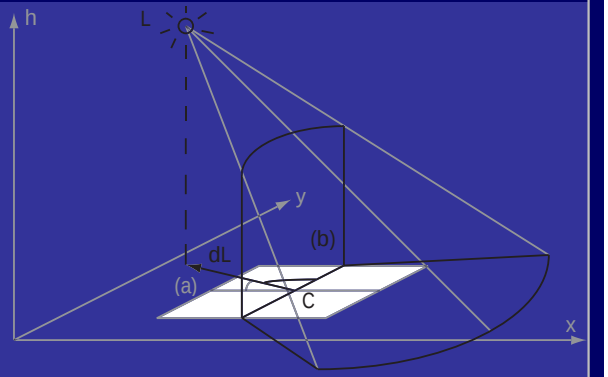
- Examples of sprites:

- Height information with little memory consumption:

## Shadow Curve Projection 1

- Shadow curves: (a) and (b)
- Sprite center: C
- Light position: $L = (x_L, y_L, h_L)$
- Direction to light $\mathbf{d}_L = (x_L, y_L) - C$








## Choosing a Shadow Curve

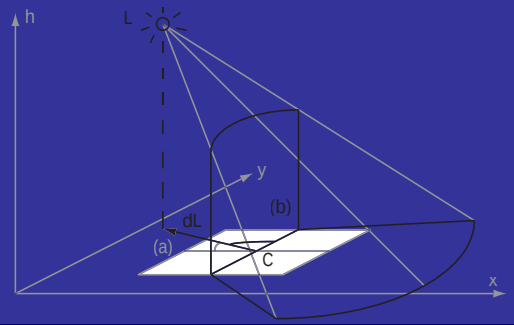
- Give each curve a direction in the xy-plane: $\mathbf{d}_{(a)}$ and $\mathbf{d}_{(b)}$
- Choose (a) if $\mathbf{d}_{(a)}$ makes the larger angle with $\mathbf{d}_L$
- Choose (b) if $\mathbf{d}_{(b)}$ makes the larger angle with $\mathbf{d}_L$





## Shadow Curve Projection

- Here curve (b) is chosen

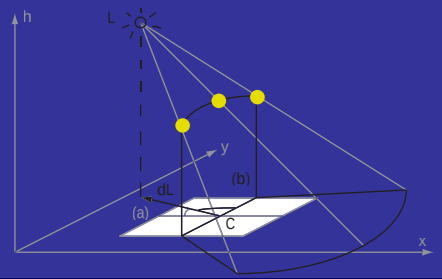






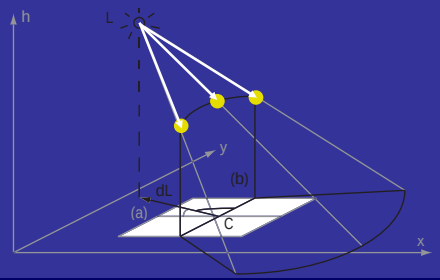

## Projecting the Curve 1/3

- Find points on the chosen curve: $P_{0,i} = (x_{0,i}, y_{0,i}, h_{0,i})$ , $i = 0,\ldots,n-1$



## Projecting the Curve 2/3

- Find a straight line from L through each curve point:
  $P_i = (x_i, y_i, h_i) = L + t_i\,(P_{0,i} - L)$

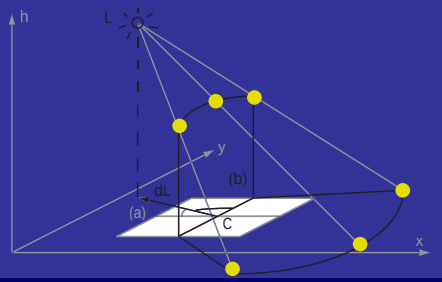







## Projecting the Curve 3/3

- $t_i$ is found where $h_i = 0$:
  $0 = h_L + t_i\,(h_{0,i} - h_L) \quad \Leftrightarrow \quad t_i = h_L \,/\, (h_L - h_{0,i})$

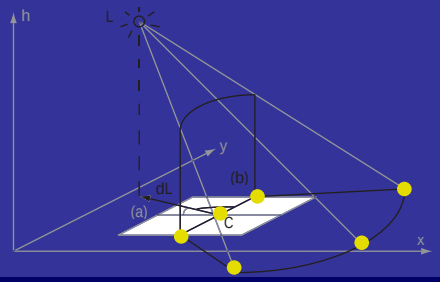







## Rendering the Shadow

- Remove height values from $P_{0,i}$ and $P_i$

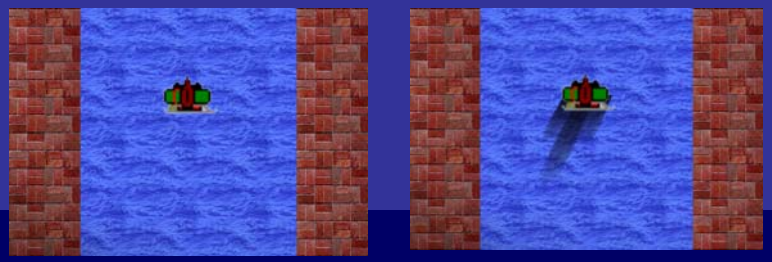







## Rendering the Shadow

- Remove height values from $P_{0,i}$ and $P_i$
- Draw a triangle strip: $P_{0,0}, P_0, P_{0,1}, P_1, \ldots, P_{0,n-1}, P_n$



## Fog from Above

- Choose a low-resolution height field

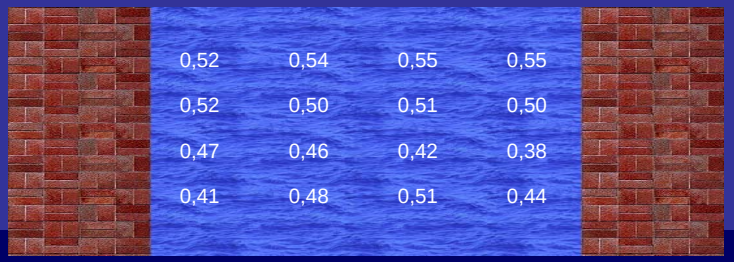
| | | | |
|---|---|---|---|
| 0,52 | 0,54 | 0,55 | 0,55 |
| 0,52 | 0,50 | 0,51 | 0,50 |
| 0,47 | 0,46 | 0,42 | 0,38 |
| 0,41 | 0,48 | 0,51 | 0,44 |





## Fog from Above

- Choose a low-resolution height field (eg. 8×8 values)
- Place the height field in your game








## Fog from Above

- Choose a low-resolution height field
- Place the height field in your game
- Find vertex positions $Q_{ij}$ for the field

| | | | |
|---|---|---|---|
| $Q_{30}$ | $Q_{31}$ | $Q_{32}$ | $Q_{33}$ |
| $Q_{20}$ | $Q_{21}$ | $Q_{22}$ | $Q_{23}$ |
| $Q_{10}$ | $Q_{11}$ | $Q_{12}$ | $Q_{13}$ |
| $Q_{00}$ | $Q_{01}$ | $Q_{02}$ | $Q_{03}$ |



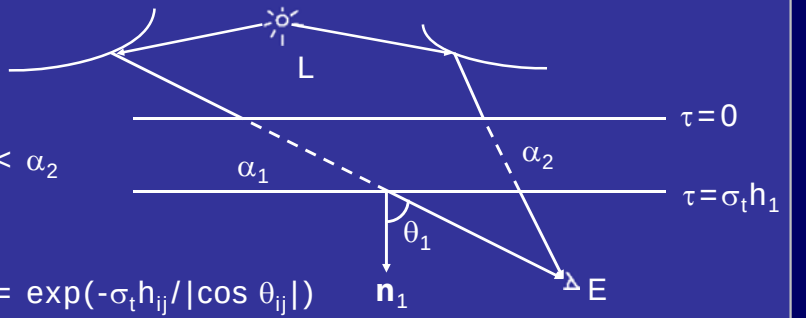


## Fog from Above

- Choose a low-resolution height field
- Place the height field in your game
- Find vertex positions $Q_{ij}$ for the field
- Find vertex normals $\mathbf{n}_{ij}$ for the field (use positions of neighbouring vertices)

# Direct Light Transmission

- Direct transmission: $\alpha \in (0,1]$




- $\alpha_1 < \alpha_2$
- $\alpha_{ij} = \exp(-\sigma_t h_{ij}/|\cos\theta_{ij}|)$





# Alpha Blending for the Fog

1.

- Choosing simplified fog shades:
  1. $(R,G,B)_{ij} = h_{ij}$ (fog in a dark place)
  2. $(R,G,B)_{ij} = 1$ (fog in a bright place)
- Choosing a blending function
  1. $L_{blend} = L_{src} + \alpha_{src}L_{dst}$
  2. $L_{blend} = (1 - \alpha_{src})L_{src} + \alpha_{src}L_{dst}$





# Alpha Blending for the Fog

2.

- Choosing simplified fog shades:
  1. $(R,G,B)_{ij} = h_{ij}$ (fog in a dark place)
  2. $(R,G,B)_{ij} = 1$ (fog in a bright place)
- Choosing a blending function
  1. $L_{blend} = L_{src} + \alpha_{src}L_{dst}$
  2. $L_{blend} = (1 - \alpha_{src})L_{src} + \alpha_{src}L_{dst}$





# Adapting the Calculations for a Phone

- OpenGL ES Common Lite profile
  - Fixed point arithmetics (GLfixed S15.16)
  - Reference: Astle & Durnil. *OpenGL ES Game Development*.
- exp(-x) look-up table (and acos(x) look-up table).
- exp(-x): 20 entries for $x \in [0,1)$,
  10 for $x \in [1,2)$, 5 for $x \in [2,3)$,
  line through $(3,e^{-3})$ and $(10,0)$ for $x \in [3,10]$,
  0 for $x > 10$

## Performance

- 400 MHz Pentium3 laptop.
- Resolution: 250×250
- Fog grid size: 16×16

| | no shadow | one strip | two strips |
|---|---|---|---|
| w/o. fog | 50.5 | 47.8 | 47.5 |
| w. fog | 30.1 | 29.0 | 28.9 |





## Conclusion


- 3D lighting effects for 2D environments
- Characteristics of described methods
  - Inexpensive calculations
  - Low memory costs
- Let's have more lighting effects in mobile games
  ... and tile-based 2D games in general





## Thank you for your attention

- Questions/comments?