Stereoscopy, From XY to Z


Samuel Gateau, NVIDIA
Robert Neuman, DISNEY
Marc Salvati, OLM

SIGGRAPH | 2011 | Vancouver
Agenda

9:00  Welcome
9:05  Stereoscopy fundamentals and depth perception  Samuel Gateau
9:40  Stereopsis and 3D hints  Marc Salvati
10:10 Depth as storytelling tool  Robert Neuman
10:30 Real-time & gaming techniques  Samuel Gateau
11:00 Cartoon Authoring for 3D  Marc Salvati
11:30 Creative choices for 3D  Robert Salvati
11:50 Managing a depth budget  Robert Salvati
12:10 Questions

Course available at
Special Thanks...
How does it work?
Changes to the rendering pipe

TWO EYES, ONE SCREEN, TWO IMAGES
Scene is viewed from one eye and projected with a perspective projection along eye direction on Near plane in Viewport.
In Stereo

Eye space

Z

Y

X

Scene

Near plane

Mono Eye

Eye space
In Stereo:

Two eyes

Left and Right eyes

Shifting the mono eye along the X axis
In Stereo:

**Two eyes**

**Left and Right eyes**

Shifting the mono eye along the X axis

Eye directions are parallels
In Stereo: Two Eyes, One Screen

Left and Right eyes
Shifting the mono eye along the X axis
Eye directions are parallels

One “virtual” screen
Left and Right eyes
Shifting the mono eye along the X axis
Eye directions are parallels

One “virtual” screen
Where the left and right frustums converge
In Stereo: Two Eyes, One Screen, Two Images

Left and Right eyes
Shifting the mono eye along the X axis
Eye directions are parallels

One “virtual” screen
Where the left and right frustums converge

Two images
2 images are generated at the near plane in each views
In Stereo: Two Eyes, One Screen, Two Images

Two images
2 images are generated at the near plane in each views
Presented independently to each eyes of the user on the real screen
Stereoscopic Rendering

Render geometry twice
From left and right eyes
Into left and right images
Basic definitions so we all speak English

DEFINING STEREOS PROJECTION
Stereo Projection

- Human vision is really like 2 eyes looking at a parallel direction
Stereo Projection

- Stereo projection matrix is a horizontally offset version of regular mono projection matrix
  - Offset Left / Right eyes along X axis
Stereo Projection

- Projection Direction is parallel to mono direction (NOT toed in)
- Left and Right frustums converge at virtual screen
Parallel, NOT Toed in!

- Historically, live camera mounted in parallel stereo would waste a lot of the view field
  - Waste view field is wasted film area
Parallel, NOT Toed in!

- Hence the Toed-in camera solution is not viable.
- But Toed in frustum introduces deformation which is really painful.
  - Image Planes are not parallel to the screen plane.
  - This can be corrected in post production but not perfect.

Eye space
Interaxial

- Distance between the 2 virtual eyes in eye space
- The mono, left & right eyes directions are all parallels
Separation

- The normalized version of interaxial by the virtual screen width
- More details in a few slides....

\[
Separation = \frac{\text{Interaxial}}{\text{Screen width}}
\]
Convergence

- Virtual Screen’s depth in eye space ("Screen Depth")
- Plane where Left and Right Frustums intersect
Parallax

- Signed Distance on the virtual screen between the projected positions of one vertex in left and right image
- Parallax is function of the depth of the vertex
Depth Perception
Where the magic happens
DEPTH PERCEPTION
Virtual vs. Real Screen

The virtual screen is perceived as the real screen.
Parallax is Depth

Real Screen

Left Image

Right Image

Virtual Screen

Scene

Virtual Space

X

Y

Z

Left Eye

Right Eye
Parallax is Depth

Parallax creates the depth perception for the user looking at the real screen presenting left and right images.
# In / Out of the Screen

<table>
<thead>
<tr>
<th>Vertex Depth</th>
<th>Parallax</th>
<th>Vertex Appears</th>
</tr>
</thead>
<tbody>
<tr>
<td>Further than Convergence</td>
<td>Positive</td>
<td>In the Screen</td>
</tr>
<tr>
<td>Equal Convergence</td>
<td>Zero</td>
<td>At the Screen</td>
</tr>
<tr>
<td>Closer than Convergence</td>
<td>Negative</td>
<td>Out of the Screen</td>
</tr>
</tbody>
</table>

**Out of the Screen**

**In the Screen**

**Convergence**

**Eye space**

**Left Eye**

**Right Eye**

**Vertex Depth**

**Parallax**

**Vertex Appears**
Equations !!!

COMPUTING PARALLAX & PROJECTION MATRIX
Computing Parallax

Thank you Thales

In eye space:

\[
\frac{\text{Parallax}_{\text{eye}}}{\text{Interaxial}} = \frac{\text{Depth} - \text{Convergence}}{\text{Depth}}
\]

\[
\text{Parallax}_{\text{eye}} = \text{Interaxial} \times \left(1 - \frac{\text{Convergence}}{\text{Depth}}\right)
\]
Computing Parallax
In image space (not pixels but in range $[0, 1]$)

In image space:

$$\text{Parallax}_{image} = \frac{\text{Parallax}_{eye}}{\text{Screen width}} = \frac{\text{Interaxial}}{\text{Screen width}} \times \left(1 - \frac{\text{Convergence}}{\text{Depth}}\right)$$

$$\text{Parallax}_{image} = \text{Separation} \times \left(1 - \frac{\text{Convergence}}{\text{Depth}}\right)$$
Computing Parallax
And clip space for free

In eye space: \[ \text{Parallax}_{\text{eye}} = \text{Interaxial} \times \left( 1 - \frac{\text{Convergence}}{\text{Depth}} \right) \]

In image space: \[ \text{Parallax}_{\text{image}} = \text{Separation} \times \left( 1 - \frac{\text{Convergence}}{\text{Depth}} \right) \]

In clip space: \[ \text{Parallax}_{\text{clip}} = 2 \times \text{Separation} \times (\text{Depth} - \text{Convergence}) \]
Parallax in normalized image space

\[ \text{Parallax} = \text{Separation} \times \left(1 - \frac{\text{Convergence}}{\text{Depth}}\right) \]

- Parallax diverges quickly to negative infinity for object closer to the eye.
- Parallax is 0 at screen depth.
- Maximum Parallax at infinity is separation ⇔ distance between the eyes.

Parallax in normalized image space vs. Depth.
Take care of your audience

REAL EYE SEPARATION
Real Eye Separation

- **Interocular** (distance between the eyes) is on average 2.5” $\cong$ 6.5 cm
- Equivalent to the visible parallax on screen for objects at infinity
- Depending on the screen width, we define a normalized “Real Eye Separation”

\[
\text{Real Eye Separation} = \frac{\text{Interocular}}{\text{Real Screen Width}}
\]

- Different for each screen model
- A reference maximum value for the Separation used in the stereo projection for a comfortable experience
Real Eye Separation is infinity

- The maximum Parallax at infinity is Separation
- Real Eye Separation should be used as the very maximum Separation value

$\text{Separation} < \text{Real Eye Separation}$
Separation must be Comfortable

- Never make the viewer look diverge
  - People don’t have the same eyes
- For Animation movie, separation must be very conservative because of the variety of the screen formats
  - IMAX vs Home theatre
- For Interactive application, let the user adjust Separation
  - When the screen is close to the user (PC scenario) most of the users cannot handle more than 50% of the Real Eye Separation
Real Eye Separation is the Maximum Parallax

\[ |\text{Parallax}| < \text{Real Eye Separation} \]
Safe Parallax Range

\[ \text{abs(Parallax)} < \text{Real Eye Separation} \]
Convergence and Separation working together

PARALLAX BUDGET
Parallax Budget
How much parallax variation is used in the frame

Parallax Budget
How much parallax variation is used in the frame

Parallax budget

Depth

Nearest pixel

Farthest pixel

Separation

Convergence
**In Screen: Farthest Pixel**

- **At 100 * Convergence, Parallax is 99% of the Separation**
  - For pixels further than 100 * Convergence, Elements looks flat on the far distance with no depth differentiation
- **Between 10 to 100 * Convergence, Parallax vary of only 9%**
  - Objects in that range have a subtle depth differentiation
Out of the Screen : Nearest pixel

- At Convergence / 2, Parallax is equal to -Separation, out of the screen
  - Parallax is very large (> Separation) and can cause eye strains
Convergence sets the scene in the screen
Defines the window into the virtual space
Defines the style of stereo effect achieved (in / out of the screen)
Separation scales the parallax budget

Scales the depth perception of the frame

Near pixel

Far pixel

Parallax

Depth

Parallax budget 1

Separation 1

Separation 2

Parallax budget 2

Convergence
Adjust Convergence

- Convergence is a Camera parameter driven by the look of the frame
  - Artistic / Gameplay decision
  - Should adjust for each camera shot / mode
    - Make sure the scene elements are in the range $[ \text{Convergence} / 2, 100 \times \text{Convergence} ]$
  - Adjust it to use the Parallax Budget properly
  - Dynamic Convergence is a bad idea
    - Except for specific transition cases
Stereopsis and 3D Hints
Depth as storytelling tool
Agenda

9:00  Welcome
9:05  Stereoscopy fundamentals and depth perception  Samuel Gateau
9:40  Stereopsis and 3D hints  Marc Salvati
10:10 Depth as storytelling tool  Robert Neuman
10:30 Real-time & gaming techniques  Samuel Gateau
11:00 Cartoon Authoring for 3D  Marc Salvati
11:30 Creative choices for 3D  Robert Salvati
11:50 Managing a depth budget  Robert Salvati
12:10 Questions

Course available at
Real-time & Gaming Techniques
Let’s do it

RENDERING IN STEREO
**Stereoscopic Rendering**

- Render geometry **twice**
- Do **stereo drawcalls**
- Duplicate **drawcalls**

From left and right **eyes**
- Apply **stereo projection**
- Modify **projection matrix**

Into left and right **images**
- Use **stereo surfaces**
- Duplicate render **surfaces**
How to implement stereo projection?

- Start from the mono transformation stack

- Inject the side, separation and convergence to get a stereo transformation stack

Stereo Projection Matrix

Stereo shift on clip position
Stereo Projection Matrix

Right handed column major matrix (OpenGL style)

- Modified version of the Projection matrix for stereo to transform geometry position from eye space to stereo clip space

\[ \text{Pos}_{\text{clip, stereo}} = \text{Projection}_{\text{stereo}} \times \text{Pos}_{\text{eye}} \]

Right handed column major matrix (OpenGL style)

\[
\text{Projection}_{\text{stereo}} = \begin{bmatrix}
    p_{11} & 0 & p_{13} - \text{side} \times \text{separation} & -\text{side} \times \text{separation} \times \text{convergence} \\
    0 & p_{22} & p_{23} & 0 \\
    0 & 0 & p_{33} & p_{34} \\
    0 & 0 & -1 & 0
\end{bmatrix}
\]

Side is -1 for left, +1 for right

\( p_{ij} \) are the coefficients of the standard mono perspective projection
Stereo Projection Matrix

Left handed row major matrix (D3D9 style)

- \( \mathbf{Pos}_{\text{clip stereo}} = \mathbf{Pos}_{\text{eye}} \times \mathbf{Projection}_{\text{stereo}} \)

**Left handed row major matrix (D3D9 style)**

\[
\mathbf{Projection}_{\text{stereo}} = \begin{bmatrix}
  p_{11} & 0 & 0 & 0 \\
  0 & p_{22} & p_{32} & 0 \\
  p_{13} + \text{side} \times \text{separation} & 0 & p_{33} & 1 \\
  -\text{side} \times \text{separation} \times \text{convergence} & 0 & p_{34} & 0
\end{bmatrix}
\]

Side is -1 for left, +1 for right

\( p_{ij} \) are the coefficients of the standard mono perspective projection
Stereo shift on clip position

- Just before rasterization in the vertex shader, offset the clip position by the parallax amount

\[ \text{clipPos.x} \; +\; \text{Side} \; \ast \; \text{Separation} \; \ast \; ( \text{clipPos.w} \; - \; \text{Convergence} ) \]

*Side is -1 for left, +1 for right*
Stereo rendering surfaces

- View dependent render targets must be duplicated
  - Back buffer
  - Depth Stencil buffer
- Intermediate full screen render targets used to process final image
  - High dynamic range, Blur, Bloom
  - Screen Space Ambient Occlusion
Mono rendering surfaces

- View independent render targets DON’T need to be duplicated
  - Shadow map
  - Spot light maps projected in the scene
How to do the stereo drawcalls?

- Simply draw the geometries twice, in left and right versions of stereo surfaces
- Can be executed per scene pass
  - Draw left frame completely
  - Then Draw right frame completely
  - Need to modify the rendering loop
- Or for each individual objects
  - Bind Left Render target, Setup state for left projection, Draw geometry
  - Bind Right render target, Setup state for right projection, Draw Geometry
  - Might be less intrusive in an engine
- Not everything in the scene needs to be drawn
  - Just depends on the render target type
## When to do what?

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Render Target Type</th>
<th>Stereo Projection</th>
<th>Stereo Drawcalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shadow maps</td>
<td>Mono</td>
<td>No Use Shadow projection</td>
<td>Draw Once</td>
</tr>
<tr>
<td>Main frame</td>
<td>Stereo</td>
<td>Yes</td>
<td>Draw Twice</td>
</tr>
<tr>
<td>Any Forward rendering pass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reflection maps</td>
<td>Stereo</td>
<td>Yes Generate a stereo reflection projection</td>
<td>Draw Twice</td>
</tr>
<tr>
<td>Post processing effect</td>
<td>Stereo</td>
<td>No No Projection needed at all</td>
<td>Draw Twice</td>
</tr>
<tr>
<td>(Drawing a full screen quad)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deferred shading lighting pass</td>
<td>Stereo G-buffers</td>
<td>Yes Be careful of the Unprojection Should be stereo</td>
<td>Draw twice</td>
</tr>
<tr>
<td>(Drawing a full screen quad)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Real-time technique
EVERYTHING IS UNDER CONTROL

What could go possibly wrong?
3D Objects

- All the 3D objects in the scene should be rendered using a unique Perspective Projection in a given frame.
- All the 3D objects must have a coherent depth relative to the scene.
- Lighting effects are visible in 3D so should be computed correctly.
  - Highlight and specular are probably best looking evaluated with mono eye origin.
  - Reflection and Refraction should be evaluated with stereo eyes.
Pseudo 3D objects: Sky box, Billboards...

- Sky box should be drawn with a valid depth further than the regular scene
  - Must be Stereo Projected
  - Best is at a very Far distance so Parallax is maximum
  - And cover the full screen
- Billboard elements (Particles, leaves) should be rendered in a plane parallel to the viewing plane
  - Doesn’t look perfect
- Relief mapping cannot be abused
Several 3D scenes

- Different 3D scenes rendered in the same frame using different scales
  - Portrait viewport of selected character
  - Split screen
- Since scale of the scene is different, Must use a different Convergence to render each scene
Out of the screen objects

- The user’s brain is fighting against the perception of hovering objects out of the screen
  - Extra care must be taken to achieve a convincing effect
- Objects should not be clipped by the edges of the window
  - Be aware of the extra horizontal guard bands
- Move object slowly from inside the screen to the outside area to give eyes time to adapt
  - Make smooth visibility transitions
  - No blinking
- Realistic rendering helps
2D Objects

- 2D object in depth attached to 3D anchor point
- Billboards in depth
- Particles with 3D positions

Starcraft2 screenshot, Courtesy of Blizzard

2D objects presenting User interface at screen
2D Objects must be drawn at a valid Depth

- With no stereo projection
  - Head Up Display interface
  - UI elements
- Either draw with no stereo projection or with stereo projection at Convergence
- At the correct depth when interacting with the 3D scene
  - Labels or billboards in the scene
  - Must be drawn with stereo projection
  - Use the depth of the 3D anchor point used to define the position in 2D window space
- Needs to modify the 2D ortho projection to take into account Stereo
2D to 3D conversion
shader function

float4 2Dto3DclipPosition(
    in float2 posClip : POSITION, // Input position in clip space
    uniform float depth          // Depth where to draw the 2D object
) : POSITION // Output the position in clip space
{
    return float4(
        posClip.xy * depth,      // Simply scale the posClip by the depth
        0,                      // Z is not used if the depth buffer is not used
        // If needed Z = ( depth * f - nf )/(f - n);
        depth );                // ( For DirectX )
    // W is the Z in eye space
}
Selection, Pointing in S3D

- Selection or pointing UI interacting with the 3D scene don’t work if drawn mono
  - Mouse Cursor at the pointed object’s depth
    Can not use the HW cursor
  - Crosshair
- Needs to modify the projection to take into account depth of pointed elements
  - Draw the UI as a 2D element in depth at the depth of the scene where pointed
  - Compute the depth from the Graphics Engine or eval on the fly from the depth buffer (Contact me for more info)
- Selection Rectangle is not perfect, could be improved
  - Cf nvidia talk at GDC 2008
Performance considerations

- At worse the frame rate is divided by 2
- But applications are rarely GPU bound so less expensive in practice
  - Since using Vsynch when running in stereo, you see the standard Vsync frequency jumps
- Not all the rendering is executed twice (Shadow maps)
- Memory is allocated twice for all the stereo surfaces
  - Try to reuse render targets when possible to save memory
- Get another GPU 😊
STEREO CULLING
3D Objects Culling

When culling is done against the mono frustum...
3D Objects Culling

... Some in screen regions are missing in the right and left frustum ...

They should be visible

Eye space

Left Eye

Mono Eye

Right Eye

Screen

Left Frustum

Mono Frustum

Right Frustum
3D Objects Culling

... And we don’t want to see out of the screen objects only in one eye ... It disturbs the stereo perception
3D Objects Culling

Here is the frustum we want to use for culling
3D Objects Culling
Computing Stereo Frustum origin offset

\[ Z = \frac{\text{Convergence}}{1 + \frac{1}{\text{Separation}}} \]
3D Objects Culling

- Culling this area is not always a good idea
- Blacking out pixels in this area is better
  - Through a shader
- Equivalent to the “Floating window” used in movies
Fetching Stereo Render Target

- When fetching from a stereo render target use the good texture coordinate
  - Render target is addressed in STEREO IMAGE SPACE
  - Use the pixel position provided in the pixel shader
  - Or use a texture coordinate computed in the vertex shader correctly
When doing deferred shading technique, Pixel shader fetch the depth buffer (beware of the texcoord used, cf previous slide):
- And evaluate a 3D clip position from the Depth fetched and XY viewport position
- Make sure to use a **Stereo Unprojection Inverse transformation** to go to Mono Eye space
- Otherwise you will be in a Stereo Eye Space !
What’s under this screen pixel?

SELECTION IN S3D
From stereo depth buffers to parallax
Aka, What’s under that cursor?

- Given the left and right depth buffers
- A pixel position in the screen (Cursor)
- How to find the unique fragment of the scene under that pixel like we would do in the mono case?
From stereo depth buffers to parallax

- There is a unique solution in mono
  - which is not trivial in stereo...
From stereo depth buffers to parallax

- The fragments are different at the Cursor position in left and right buffer
From stereo depth buffers to parallax

- Correct left and right cursor locations
  - Are pointing at the same scene fragment
  - Are shifted away from the mono position from **Parallax**
Parallax is bounded in a given range of pixels \([MinParallax, MaxParallax]\)
- Deduced From the range \([near, far]\)
- So we know where to look in the depth buffers
- Correct location for the left & right pixels is in the neighborhood of the mono pixel
- Now we need a technique to find the correct solution in left and right depth buffers in this area
From stereo depth buffers to parallax

- Search area in each buffer is only half of the total parallax range and symmetrical around the mono pixel
- Look into pixel segment from the depth buffers
From stereo depth buffers to parallax

- The left and right pixels over the same scene fragment
  - Are horizontally at the same distance away from the mono pixel because they should be shifted by the same half parallax
  - And the 2 depths found should be equal and evaluate to the correct half parallax
From stereo depth buffers to parallax

- Start search from the mono pixel
- Progress on both sides pixel by pixel to find the one where

$$\text{Parallax}(depth) = X_{\text{offset}}$$
$$X_{\text{offset}}_{\text{left}} = X_{\text{offset}}_{\text{right}}$$
From stereo depth buffers to parallax

- Min parallax could be negative
  - Scene out of the screen
- Look into both directions around the mono pixel
**From stereo depth buffers to parallax**

- Start search from the mono pixel
- Progress on both side pixel by pixel to find the one where
  - Parallax( Depth ) / 2 = Xoffset
  - Xoffset Right
From stereo depth buffers to parallax

- Start search from the mono pixel

![Diagram showing stereo depth buffers and parallax search](image-url)
From stereo depth buffers to parallax

- Progress pixel by pixel to find the one where
- \( \text{Parallax( depth )} / 2 = \text{X offset} \)
Cartoon authoring for 3D
Creative choices for 3D
Managing a depth budget
Special Thanks...
Questions

Ping us for any question at sgateau@nvidia.com