Exercise 1

Stereoscopy

General Exercise Organization

Groups

- Solve the exercise in teams of 2-3 students (the teams will stay fixed throughout the semester [exceptional circumstances notwithstanding])

Passing Criteria

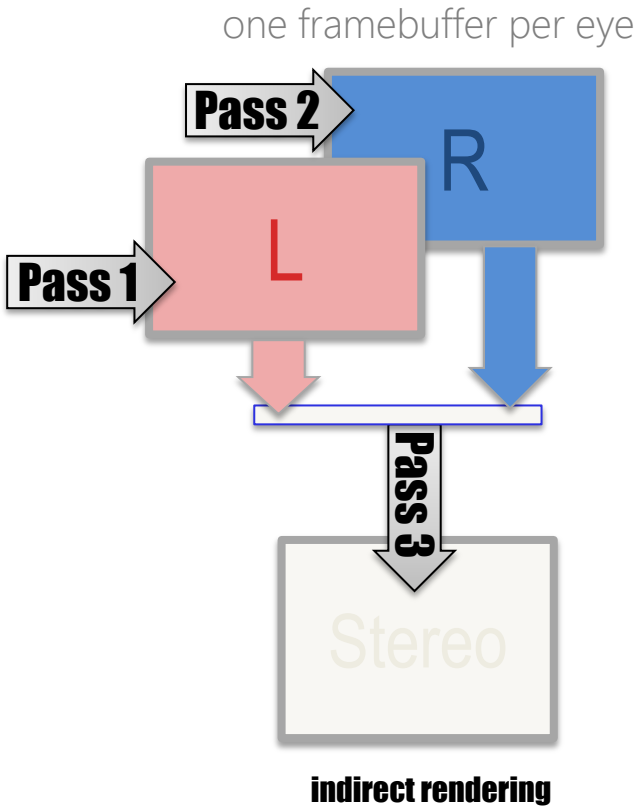
- Passing the exercise grants you full credits upon passing the exam - otherwise, you will only receive half
- Exercises do not factor into your final grade - only the examination is being graded

- In order to pass the exercise, you need:
  - 50% of total points from all exercises 1-4
  - at least 2 points in exercises 1-4 each
- Exercise 0 is fully optional and awards you bonus points
- Bonus points work towards your 50% goal!

Exercise 1 – Stereo

Multi Pass Rendering

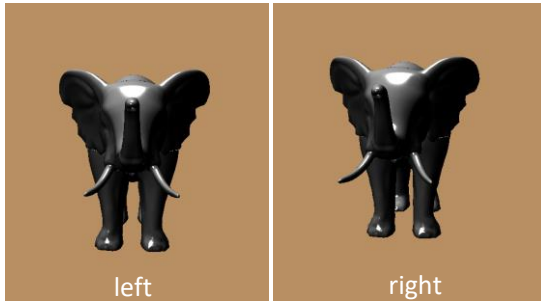
- One framebuffer per eye - each with color and depth texture
- Corresponding Indices:
  - 0 .. Left
  - 1 .. Right
- Final shader program combining the two views

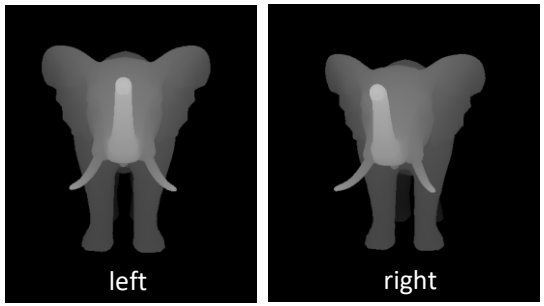


Exercise 1 – Stereo

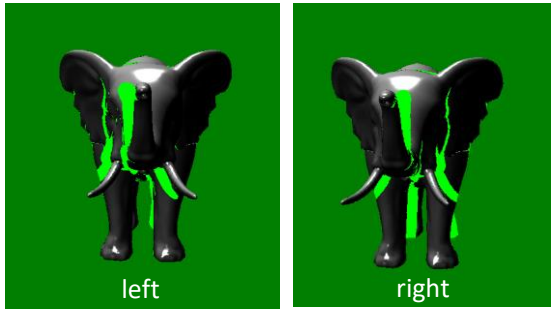
Finalize fragment shader

- Input





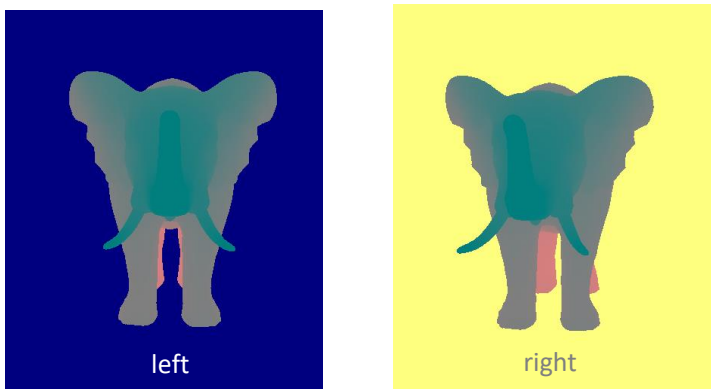
- Remapped



Exercise 1 – Stereo

Finalize fragment shader

- Parallax



- Anaglyph

Stereo Rendering – Indirect Mode

Computer Graphics and Visualization

```
protected:
    texture      col_texs[2], dep_texs[2], rnd_tex;
    frame_buffer fbos[2];
    shader_program finalize_prog;
public:
    stereo() : node("stereo"), col_texs{ "[R,G,B]", "[R,G,B]" },
               dep_texs{ "[D]", "[D]" }, rnd_tex("[R,G,B]")
    {
        for (int i = 0; i < 2; ++i) {
            col_texs[i].set_mag_filter(TF_NEAREST);
            col_texs[i].set_min_filter(TF_NEAREST);
            dep_texs[i].set_mag_filter(TF_NEAREST);
            dep_texs[i].set_min_filter(TF_NEAREST);
    }   }
    void init_frame(context& ctx)
    {
        unsigned w = ctx.get_width(), h = ctx.get_height();
        if (!fbos[0].is_created()) {
            for (int i = 0; i < 2; ++i) {
                col_texs[i].set_width(w); col_texs[i].set_height(h);
                col_texs[i].create(ctx);
                dep_texs[i].set_width(w); dep_texs[i].set_height(h);
                dep_texs[i].create(ctx);
                fbos[i].create(ctx, w, h);
                fbos[i].attach(ctx, dep_texs[i]);
                fbos[i].attach(ctx, col_texs[i]);
                if (!fbos[i].is_complete(ctx)) {
                    std::cerr << "ups should be complete!" << std::endl;
    ;
```

declaration of color & depth textures, framebuffer object and shader program

Initialization of texture format and disabling of texture filtering

creation of textures and fbo as GPU objects based on render context ctx

textures are attached to fbo who is then checked for completeness

Stereo Rendering – Indirect Mode

Computer Graphics and Visualization

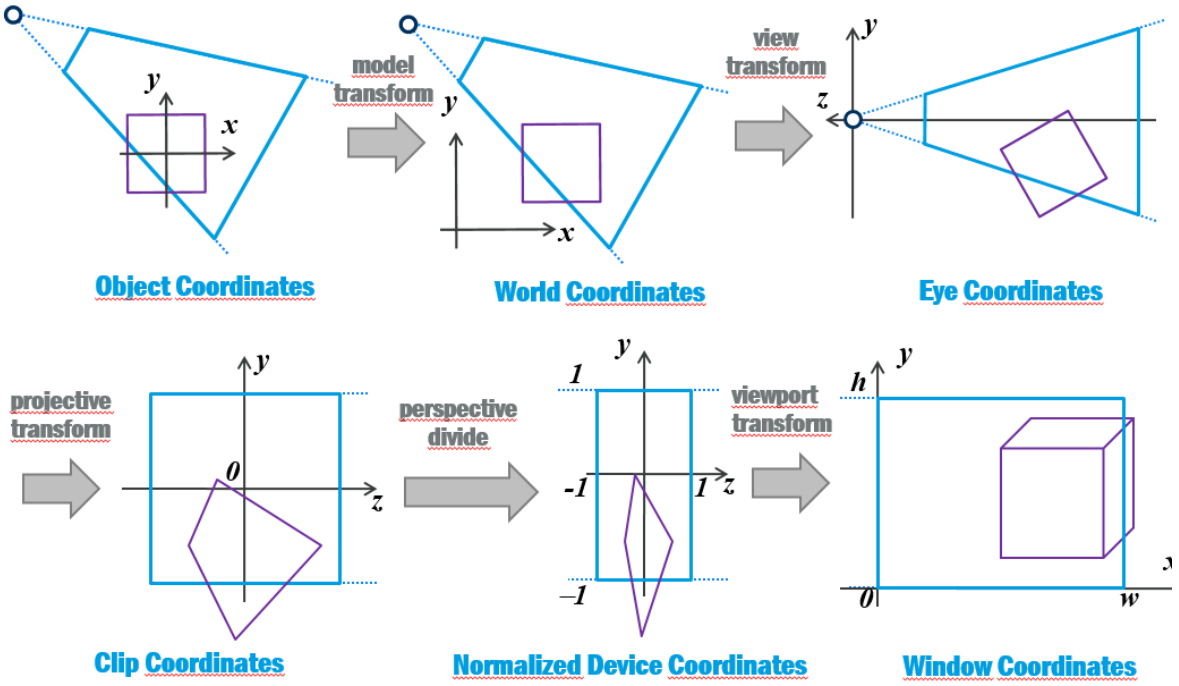
```
void indirect_two_pass_stereo(context& ctx)
{
   mat4 MVP[2];
   // render once per eye
   for (int i = 0; i < 2; ++i) {
      // enable rendering to fbo of eye
      fbos[i].enable(ctx);
      // clear framebuffer and setup modelview projection of eye
      :
      // store per eye modelview projection matrix
      MVP[i] = ctx.get_projection_matrix()*ctx.get_modelview_matrix();
      // render scene
      render_scene(ctx);
      // recover previous state
      fbos[i].disable(ctx);
   }
   mat4 iMVP[2] = { inv(MVP[0]), inv(MVP[1]) };
   col_texs[0].enable(ctx, 0); finalize_prog.set_uniform(ctx, "col_tex_0", 0);
   dep_texs[0].enable(ctx, 2); finalize_prog.set_uniform(ctx, "dep_tex_0", 2);
   :
   finalize_prog.set_uniform(ctx, "MVP_1", MVP[0]);
   finalize_prog.set_uniform(ctx, "iMVP_1", iMVP[0]);
   :
   glDisable(GL_DEPTH_TEST);
   render_screen_filling_quad(ctx, finalize_prog);
   glEnable(GL_DEPTH_TEST);
   col_texs[0].disable(ctx);
   dep_texs[0].disable(ctx);
   :
```

One offline render pass per eye. Modelview projection matrices are memorized

Invert matrices, enable textures and pass uniforms to program

Render screen filling quad ignoring depth test. Here the finalize_prog is used to combine the two images in a single stereo frame

Transformation Matrices



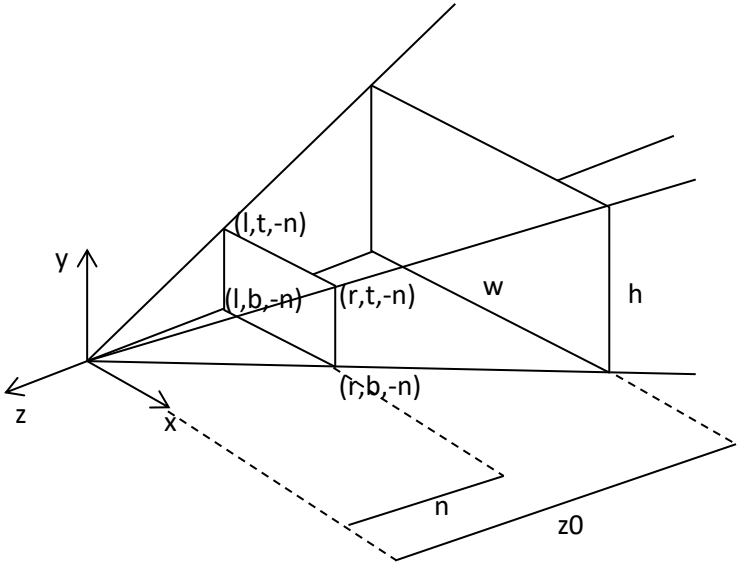
Transformation Matrices

Projection Matrix – View Frustrum:

- Points in **eye coordinates** are projected to near clipping plane with the following matrix:

$$\boldsymbol{P}_{\text{frustum}}(l,r,b,t,n,f) = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

  - l, r … left and right on near clipping plane
  - t, b … top and bottom on near clipping plane
  - n, f … depth of near and far clipping planes
- In order to calculate the parameter use the intercept theorem $e.g.\ t = \frac{1}{2}h * \frac{n}{z0}$



Transformation Matrices

Projection Matrix Stereo Case:

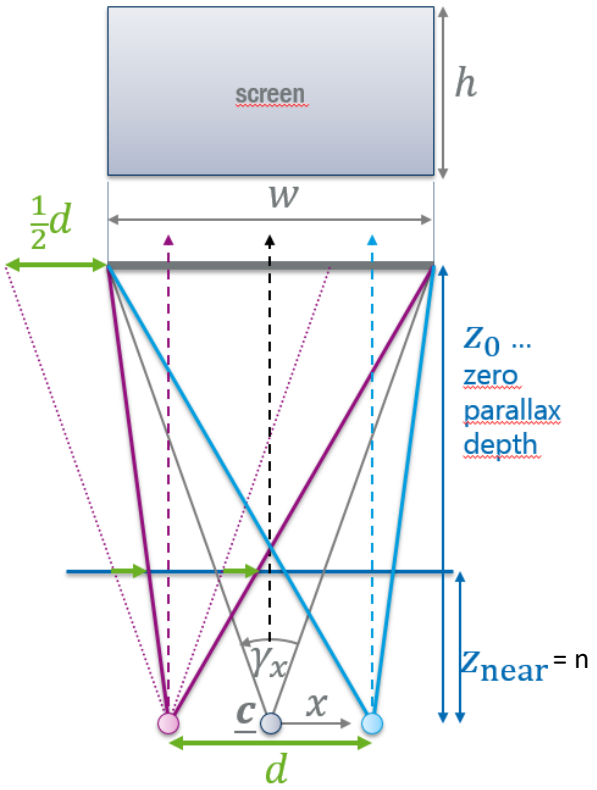
- For the stereo case we have view frusti for left / right eye which are
  - translated by $\mp ½d$ and
  - sheared by $\pm ½d/z_0$

Remarks on matrix calculations:

- t and b can be computed as before since the changes only takes place for the x-coordinate
- l and r has to be translated by a delta:

$$\pm\frac{1}{2}d * w * \frac{n}{z0}$$

- The transformation matrix $\boldsymbol{T}_{\pm}$ translates the eye point by $\pm\frac{1}{2}d * w$ in x-direction taking shearing into account



Exercise 1 – Stereo

Transformation Matrices
Effect on lighting

- **per eye**

$$\boldsymbol{MV}_{\pm} = \boldsymbol{T}_{\pm} \cdot \boldsymbol{MV}$$

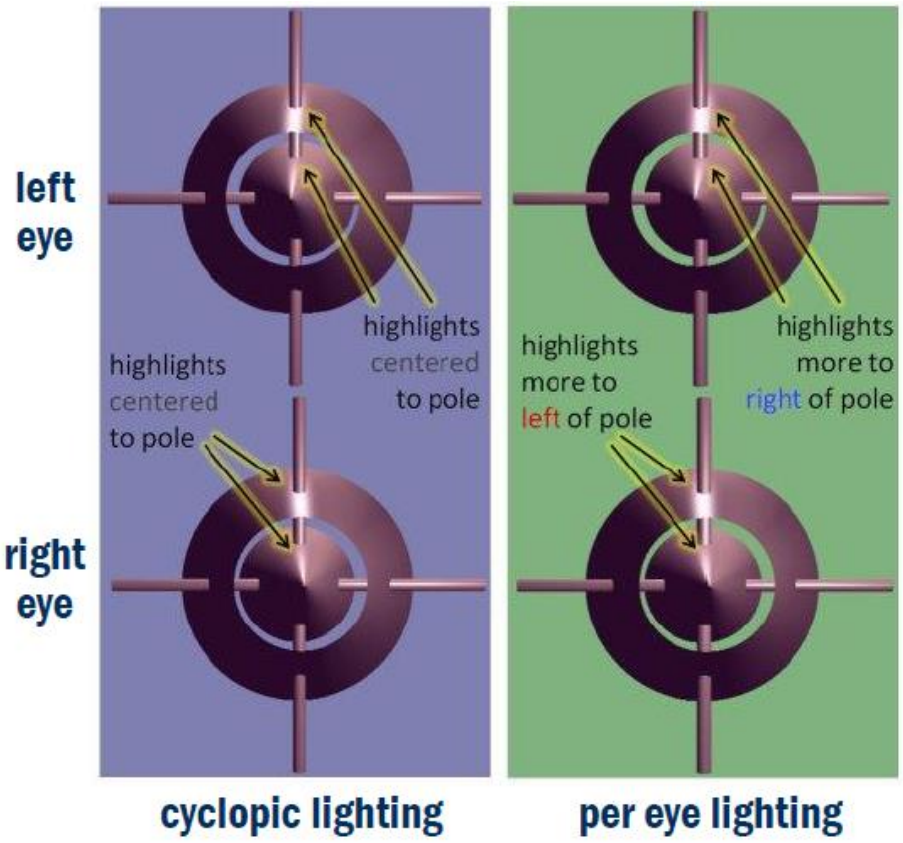
$$\boldsymbol{MVP}_{\pm} = \boldsymbol{P}_{\text{frustum},\pm}(l, r, b, t, n, f) \cdot \boldsymbol{MV}_{\pm}$$

- **cyclopic**

$$\boldsymbol{P}_{\pm} = \boldsymbol{P}_{\text{frustum},\pm}(l, r, b, t, n, f) \cdot \boldsymbol{T}_{\pm}$$

$$\boldsymbol{MVP}_{\pm} = \boldsymbol{P}_{\pm} \cdot \boldsymbol{MV}$$

(subscript −/+ for left/right eye)



Exercise 1 – Stereo

Anaglyph

- True Anaglyph

$$\begin{pmatrix} r_a \\ g_a \\ b_a \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0{,}299 & 0{,}587 & 0{,}114 \end{pmatrix} \cdot \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix}$$

- Gray Anaglyph

$$\begin{pmatrix} r_a \\ g_a \\ b_a \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}299 & 0{,}587 & 0{,}114 \end{pmatrix} \cdot \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix}$$

http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx

Exercise 1 – Stereo

Anaglyph

- Color Anaglyph

$$\begin{pmatrix} r_a \\ g_a \\ b_a \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix}$$

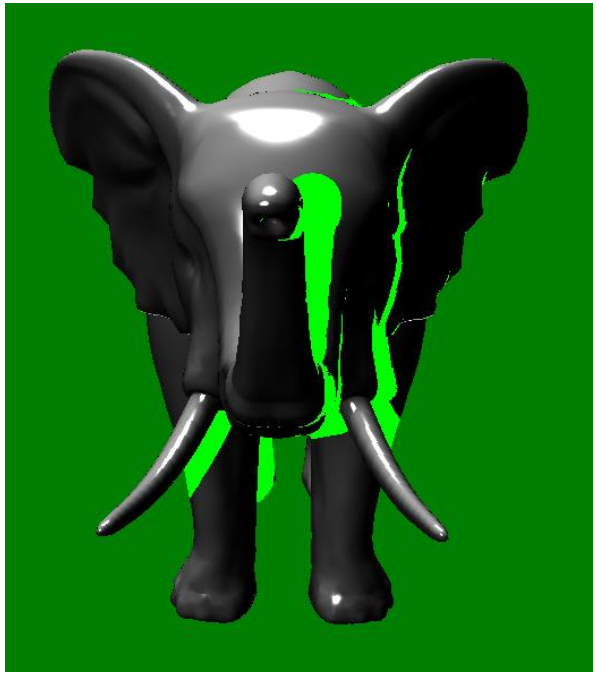
- Half Color Anaglyph

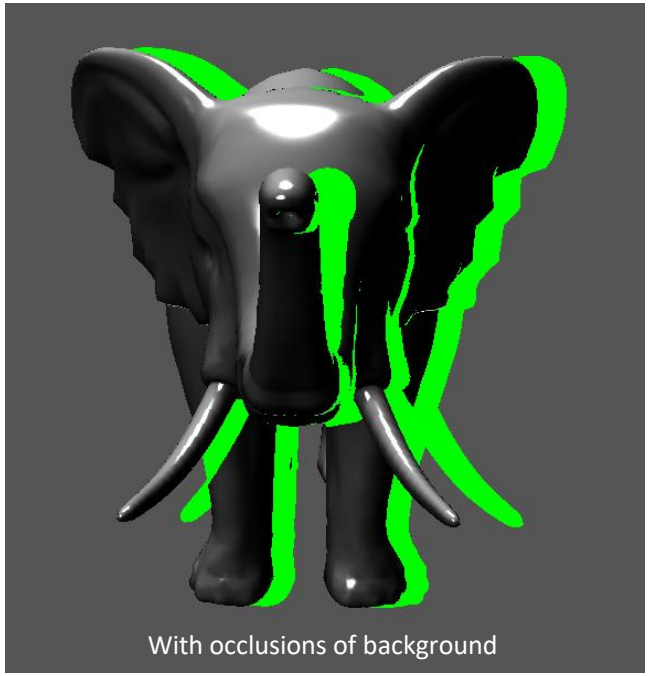
$$\begin{pmatrix} r_a \\ g_a \\ b_a \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix}$$

http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx

Remapping

- Making occlusions visible by using color that the other eye sees





Right view with color texture of left view showing that regions that cannot be seen by the left eye

Stereo Rendering – Remapping

pixel & tex coords

$\underline{x} = (x, y)$ $\underline{\tau} = (u, v)$

colors and depths at $\underline{x}$

$\ddot{c}_0, \ddot{c}_1$

d_0, d_1

surface points seen at $\underline{x}$ in each image

$\underline{p}_0, \underline{p}_1$

remapping surface points into other image by using texture coordinates

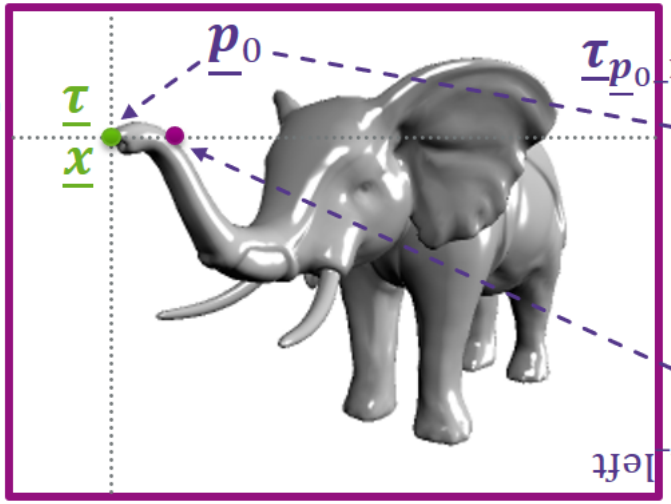
$\underline{\tau}_{p_0\_in\_right}$

$\underline{\tau}_{p_1\_in\_left}$

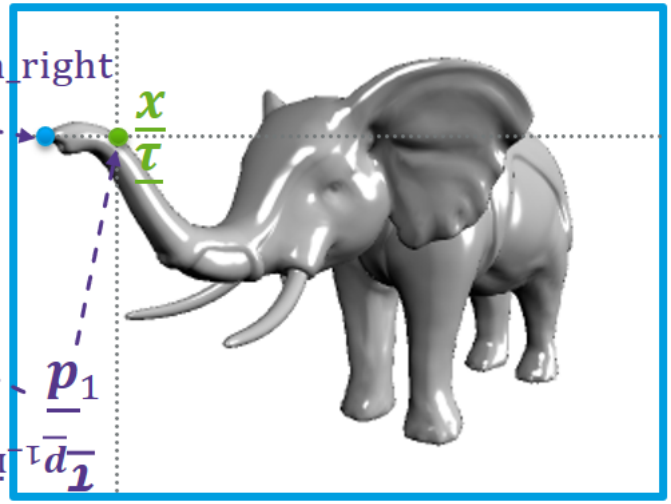
resulting in corresponding colors and depths from other image

$\ddot{c}_{1\_in\_left}, d_{1\_in\_left}$

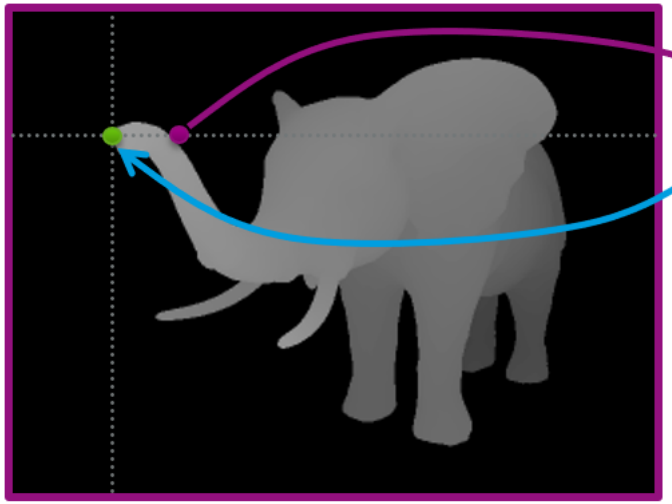
$\ddot{c}_{0\_in\_right}, d_{0\_in\_right}$



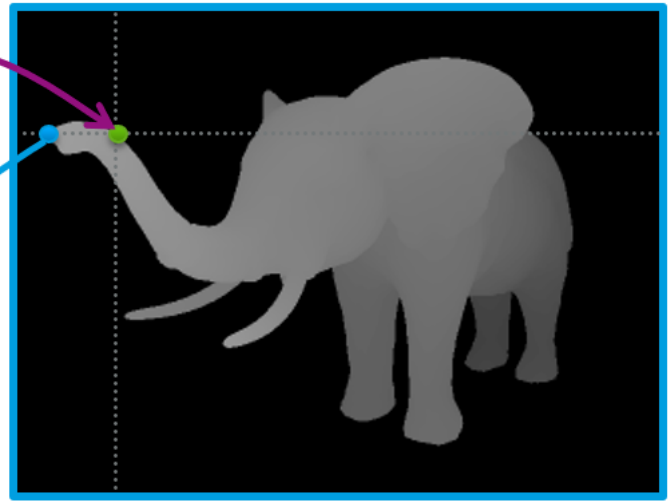
left eye color image



right eye color image



left eye depth image



right eye depth image

Stereo Rendering – Remapping

- For remapping one needs to compute texture coordinates $\underline{\tau}_{\underline{p}_0\_\text{in\_right}}$, which is achieved as follows:
  - extract window coordinates of $\underline{p}_0$ from `gl_FragCoord.xy` and depth value from left eye depth image
  - Transport $\underline{p}_0$ back to world coordinates with inverse modelview projection matrix ($\boldsymbol{MVP}_l$) of left eye
  - Transform $\underline{p}_0$ to clip coordinates of right eye with $\boldsymbol{MVP}_r$
  - Perform w-clip and extract texture coordinates from normalized device coordinates
- Finally, use $\underline{\tau}_{\underline{p}_0\_\text{in\_right}}$ to lookup corresponding color / depth values in other textures

$$\begin{pmatrix} x_{window} \\ y_{window} \\ z_{window} \end{pmatrix} = \begin{pmatrix} \frac{w}{2}(x_{NDC}+1)+x \\ \frac{h}{2}(y_{NDC}+1)+y \\ \frac{1}{2}(z_{NDC}+1) \end{pmatrix}$$

slide 72

`vec3(gl_FragCoord.xy, texture(depth_l,texcrd))`

$$\begin{pmatrix} x_{window,l} \\ y_{window,l} \\ z_{window,l} \end{pmatrix}$$

$$\begin{pmatrix} x_{NDC,l} \\ y_{NDC,l} \\ z_{NDC,l} \end{pmatrix} = \begin{pmatrix} \frac{2}{w}(x_{window,l}-x)-1 \\ \frac{2}{h}(y_{window,l}-y)-1 \\ 2z_{window,l}-1 \end{pmatrix}$$

$$\begin{pmatrix} x_{clip,r} \\ y_{clip,r} \\ z_{clip,r} \\ w_{clip,r} \end{pmatrix} = \boldsymbol{MVP}_r \boldsymbol{MVP}_l^{-1} \begin{pmatrix} x_{NDC,l} \\ y_{NDC,l} \\ z_{NDC,l} \\ \mathbf{1} \end{pmatrix}$$

$$\begin{pmatrix} x_{NDC,r} \\ y_{NDC,r} \\ z_{NDC,r} \end{pmatrix} = \frac{1}{w_{clip,r}} \begin{pmatrix} x_{clip,r} \\ y_{clip,r} \\ z_{clip,r} \end{pmatrix}$$

$$\underline{\tau}_{\underline{p}_0\_\text{in\_right}} = \frac{1}{2} \begin{pmatrix} x_{NDC,r}+1 \\ y_{NDC,r}+1 \end{pmatrix}$$

Visibility Check

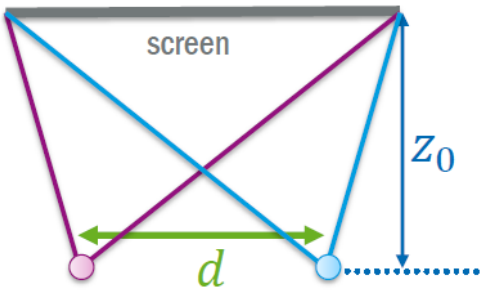
- Check if surface point $\underline{\boldsymbol{p}}_0$ seen in left image is visible in right image: $d_{1\_\text{in\_left}} = depth_1(\underline{\boldsymbol{\tau}}_{\underline{\boldsymbol{p}}_0\_\text{in\_right}}) \approx d_0$

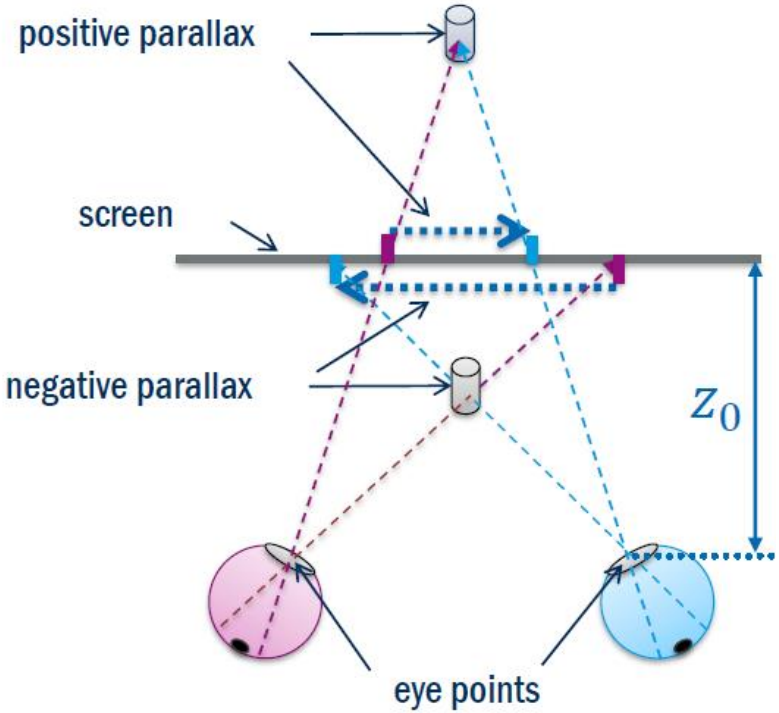
  If violated, only $d_{1\_\text{in\_left}} < d_0$ possible.

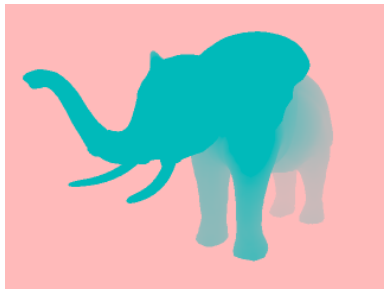
Parallax and Eye Seperation

- Distance between projection points is called parallax
- Parallax p depends on depth z of object, screen depth z_0 (distance of screen wrt eye points) and eye separation d:

$$p(z) = d \cdot (1 - z_0/z)$$







parallax mapped to color
(blue negative, red positive)