Shadow Mapping

Alba Navarro Rosales

January 27, 2021

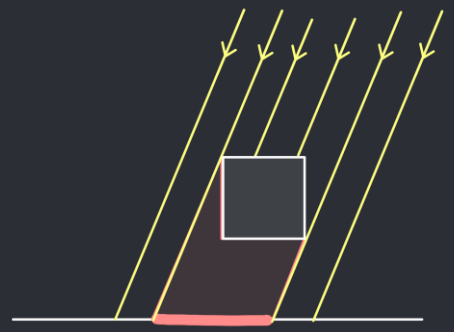
Churchill College CompSci Talks

Outline

- Lights & Shadows
- What is Shadow Mapping?
- OpenGL rendering pipeline overview
- Method
- Artefacts that occur
- Demo time! :)
- Shadow Mapping in context
- Summary – key takeaways

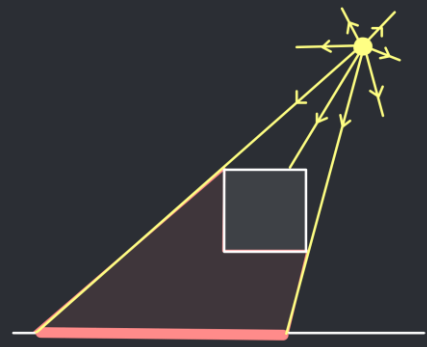
Lights & shadows

- Shadows…
  - Add realism
  - Convey depth
  - Convey spatial relationships between objects
- Different lights cast different shadows
  - Directional Light:



Orthographic Projection

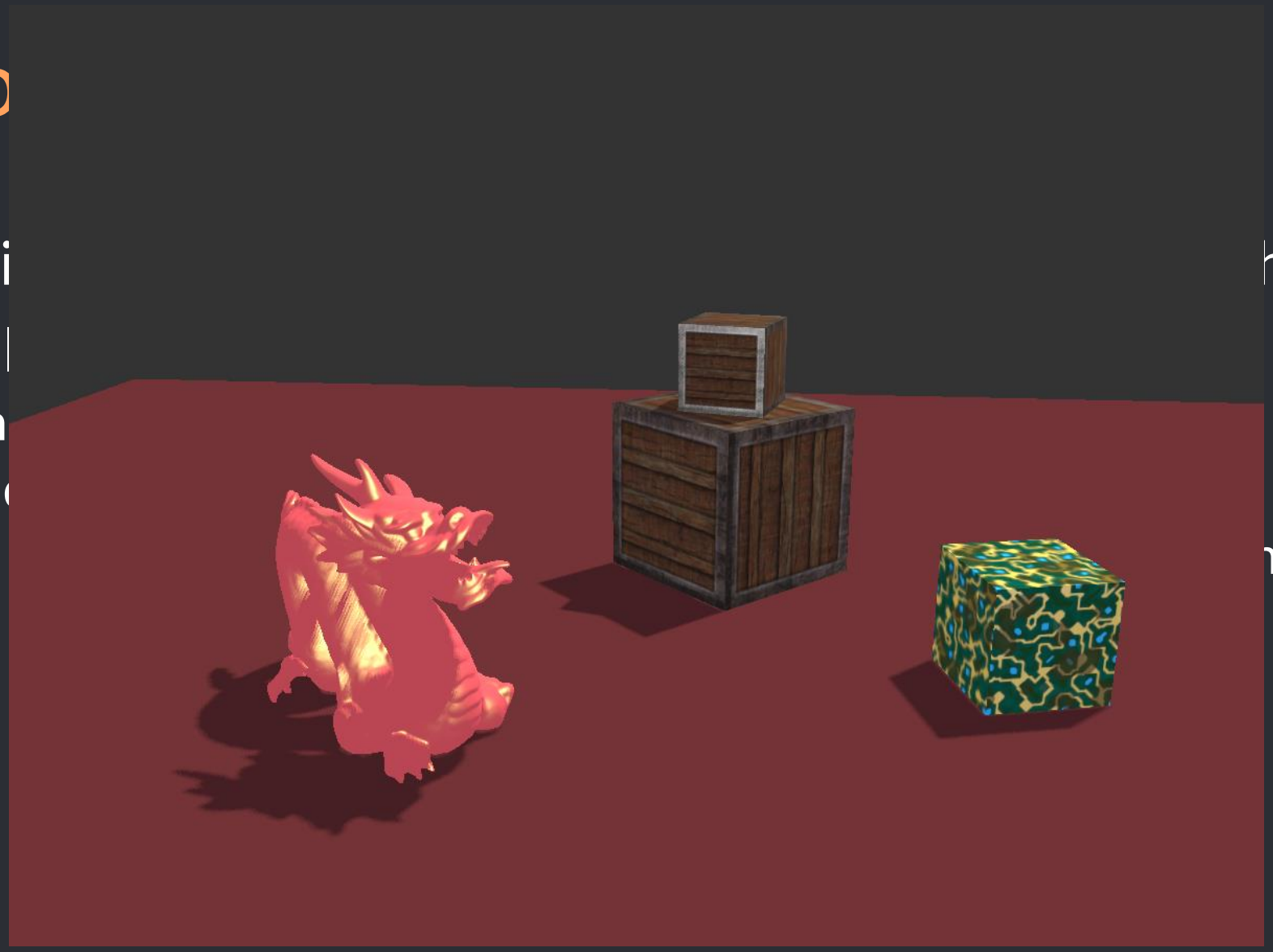
  - Point Light:



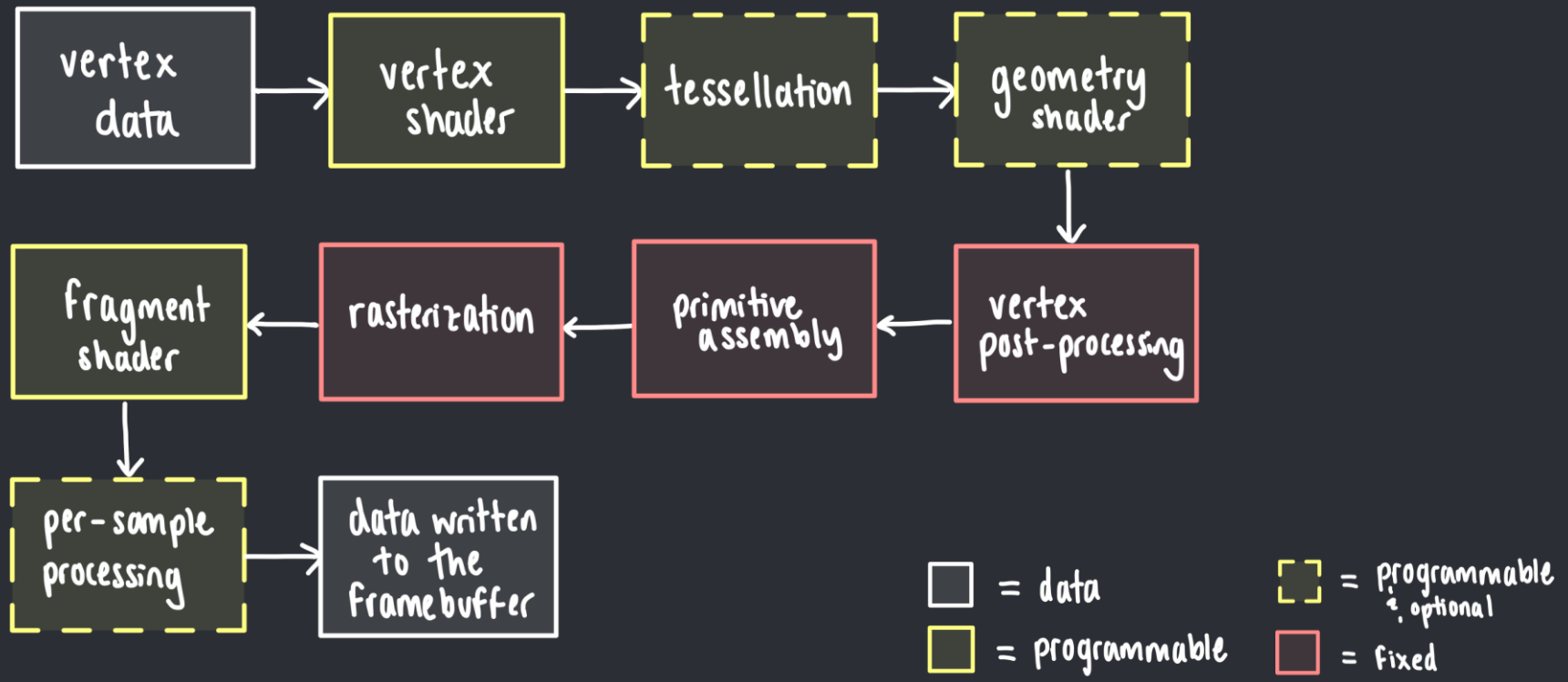
Perspective Projection

Shado

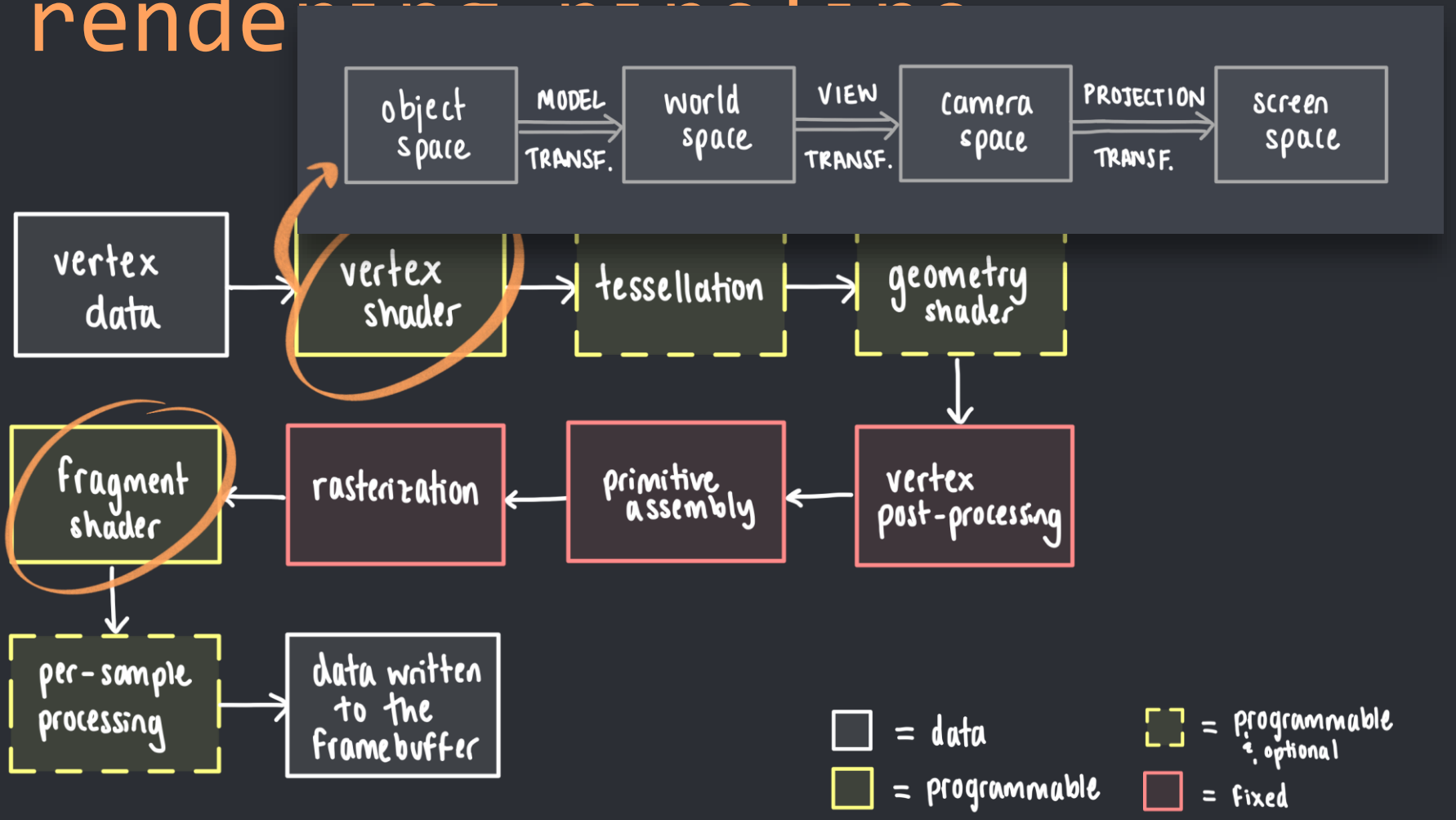
- A techni hics
- 2 pass al
  1. Gen
  2. Ren nt is lit



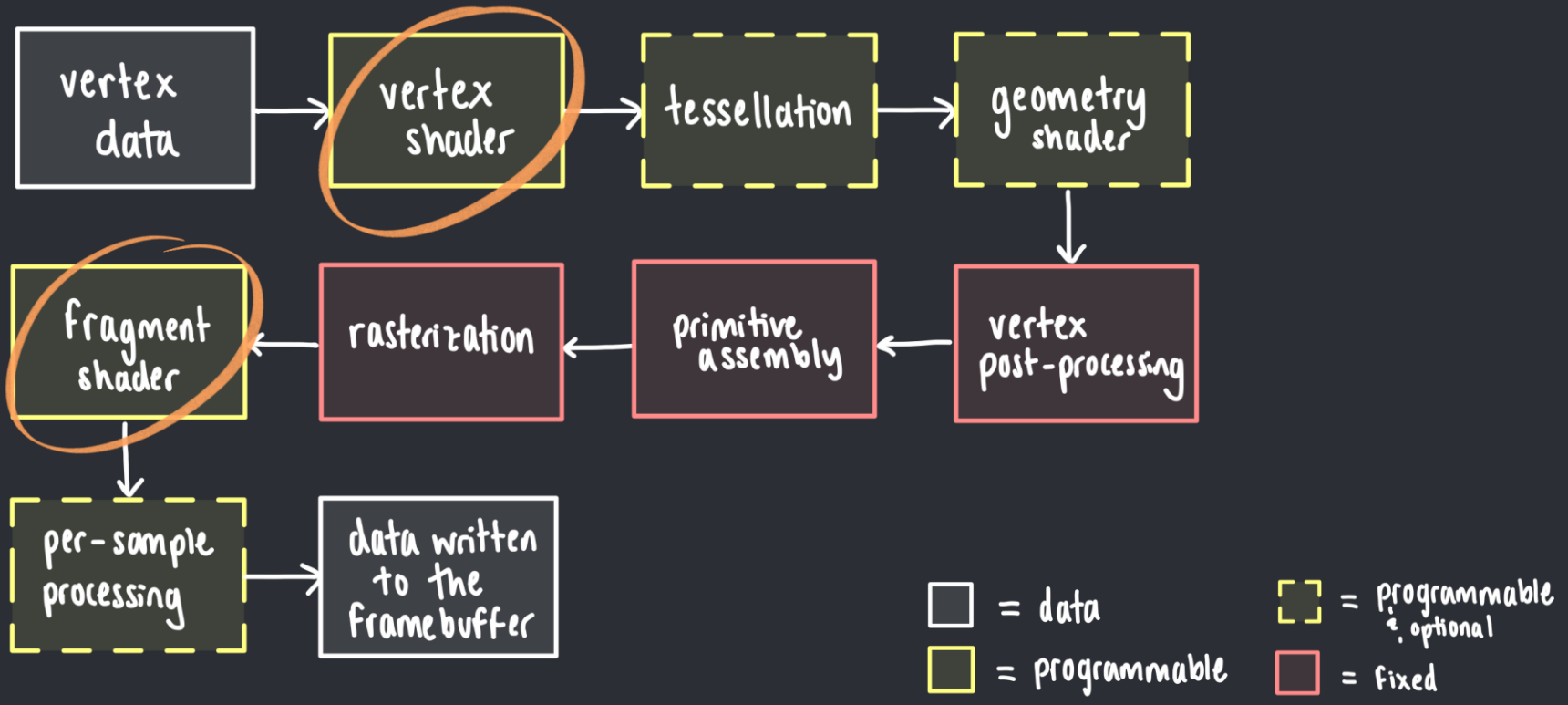
OpenGL rendering pipeline



OpenGL rendering pipeline



OpenGL rendering pipeline



Method (directional lights)

- 2 pass algorithm:
  1. Generate Depth Map by rendering scene from light's POV
  2. Render scene from camera's POV…

     …using generated Depth Map to determine if fragment is lit or in shadow

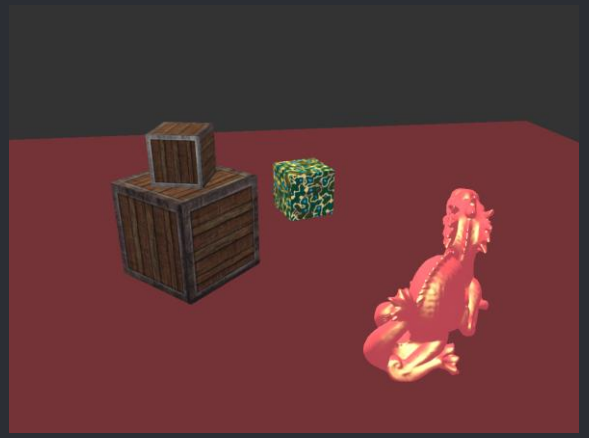
Method (directional lights)

- 2 pass algorithm:
  1. Generate Depth Map by rendering scene from light's POV
  2. Render scene from camera's POV…

     …using generated Depth Map to determine if fragment is lit or in shadow

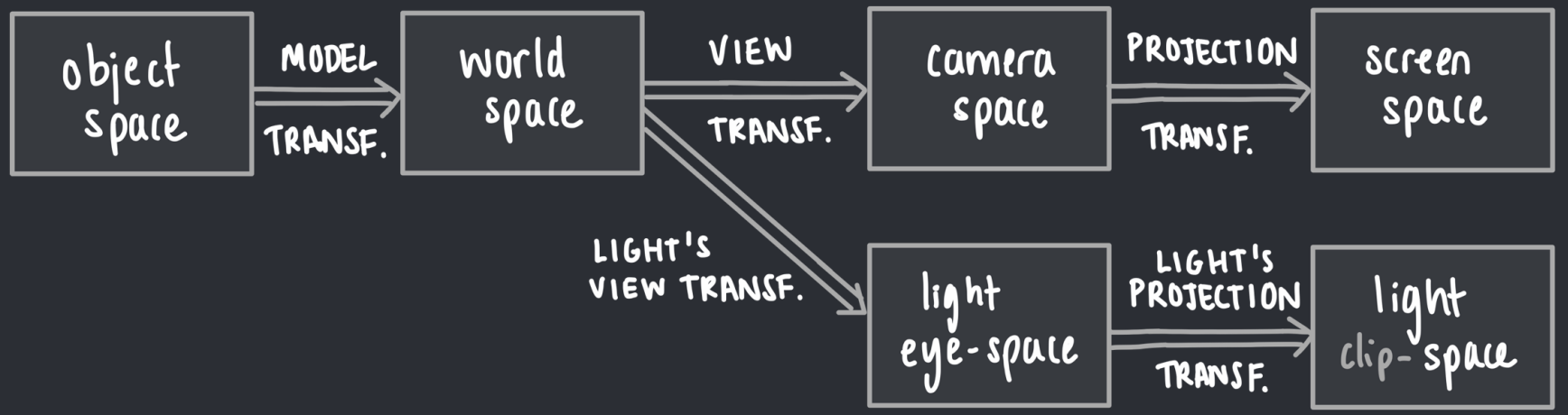
Step 1) generate depth map (1/2)

- Render scene from light's POV
- Record depth values only (not colour)
  - Depth values stored in a depth buffer - This is our depth map!
  - So, depth map stores depth of the closest fragments as seen from the light's perspective



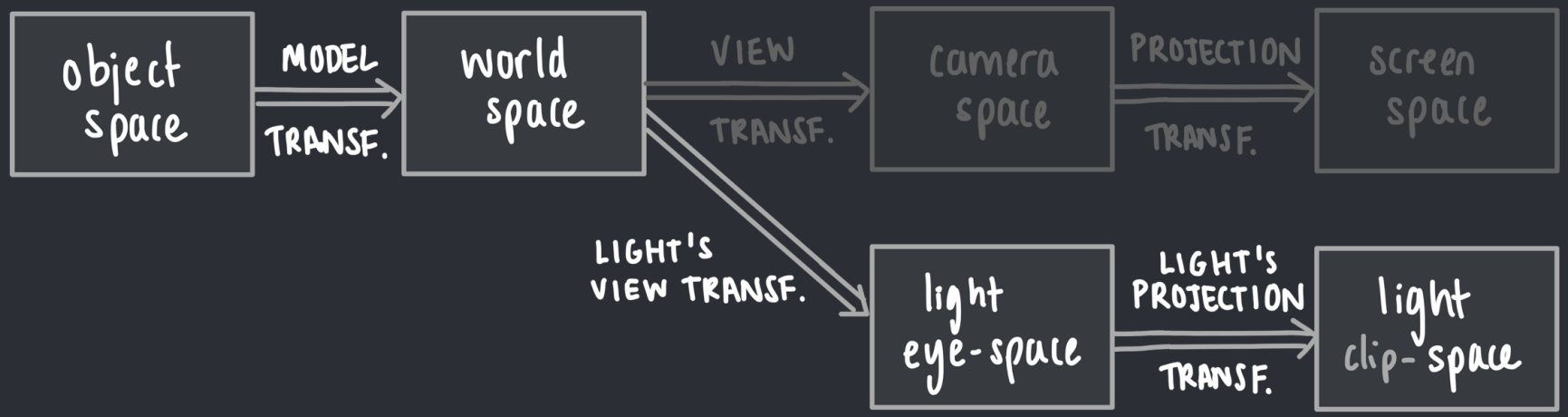
Step 1) generate depth map (2/2)

1. Create a texture object → our depth map.
2. Transform scene to light space



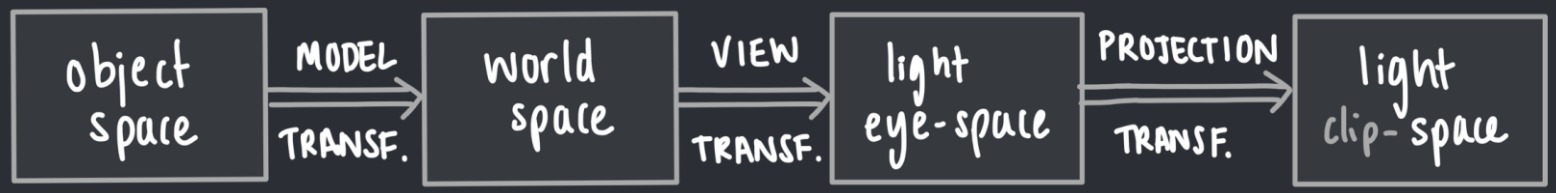
Step 1) generate depth map (2/2)

1. Create a texture object → our depth map.
2. Transform scene to light space



Step 1) generate depth map (2/2)

1. Create a texture object → our depth map.
2. Transform scene to light space



   - Use view & projection matrices specific to the light source
   - Then, calculate an MVP matrix to use in the vertex shader
3. Render to depth map
   - Vertex shader – transforms vertices to light space using calculated matrix
   - Fragment shader – empty, since no colour data

Method (directional lights)

- 2 pass algorithm:
  1. Generate Depth Map by rendering scene from light's POV
  2. Render scene from camera's POV…

     …using generated Depth Map to determine if fragment is lit or in shadow

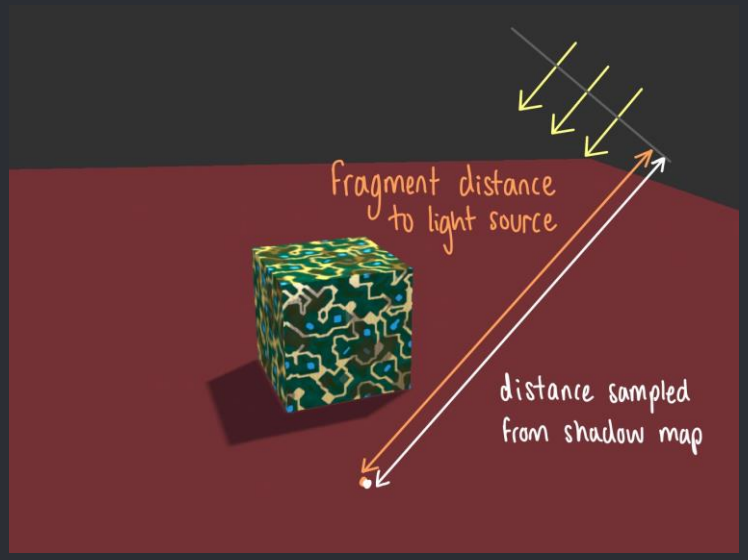
Step 2) render scene (1/2)

- Render scene as usual, from camera's POV
  - Vertex shader
    - transforms vertices to screen space (MVP matrix)
    - also transforms them to light space → to be used in fragment shader *
  - Fragment shader
    - determines colour of fragment (e.g. Phong)
    - checks if lit or in shadow:
      - Samples corresponding point in depth map → depth of closest object to light, z_c
      - Compares with depth of fragment in light space (*), z_f
      - If $z_f > z_c$ ⇒ in shadow (not seen from light)

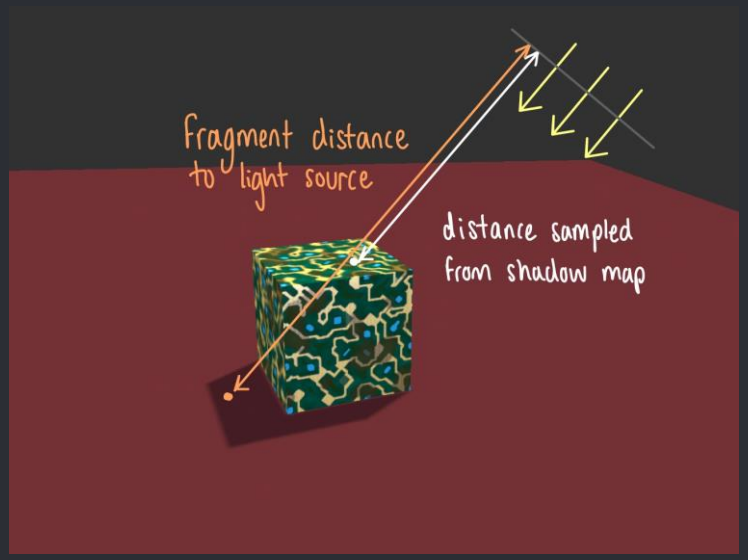
Step 2) render scene (2/2)

- Fragment shader – checks if lit or in shadow:
  - Samples corresponding point in depth map → depth of closest object to light, z_c
  - Compares with depth of fragment in light space, z_f
  - If $z_f > z_s \Rightarrow$ in shadow (not seen from light)

e.g.1. Point not in shadow

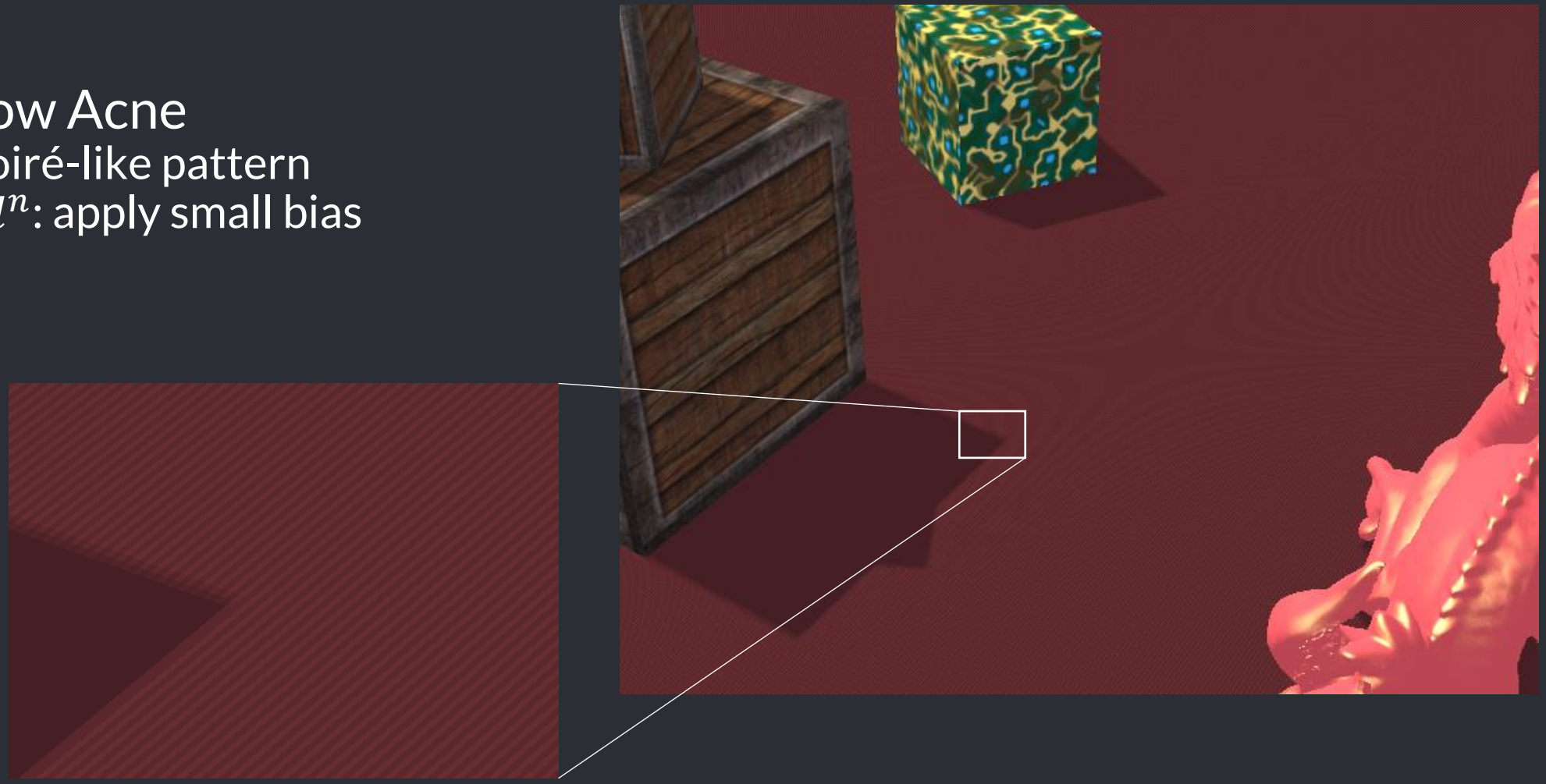


e.g.2. Point in shadow



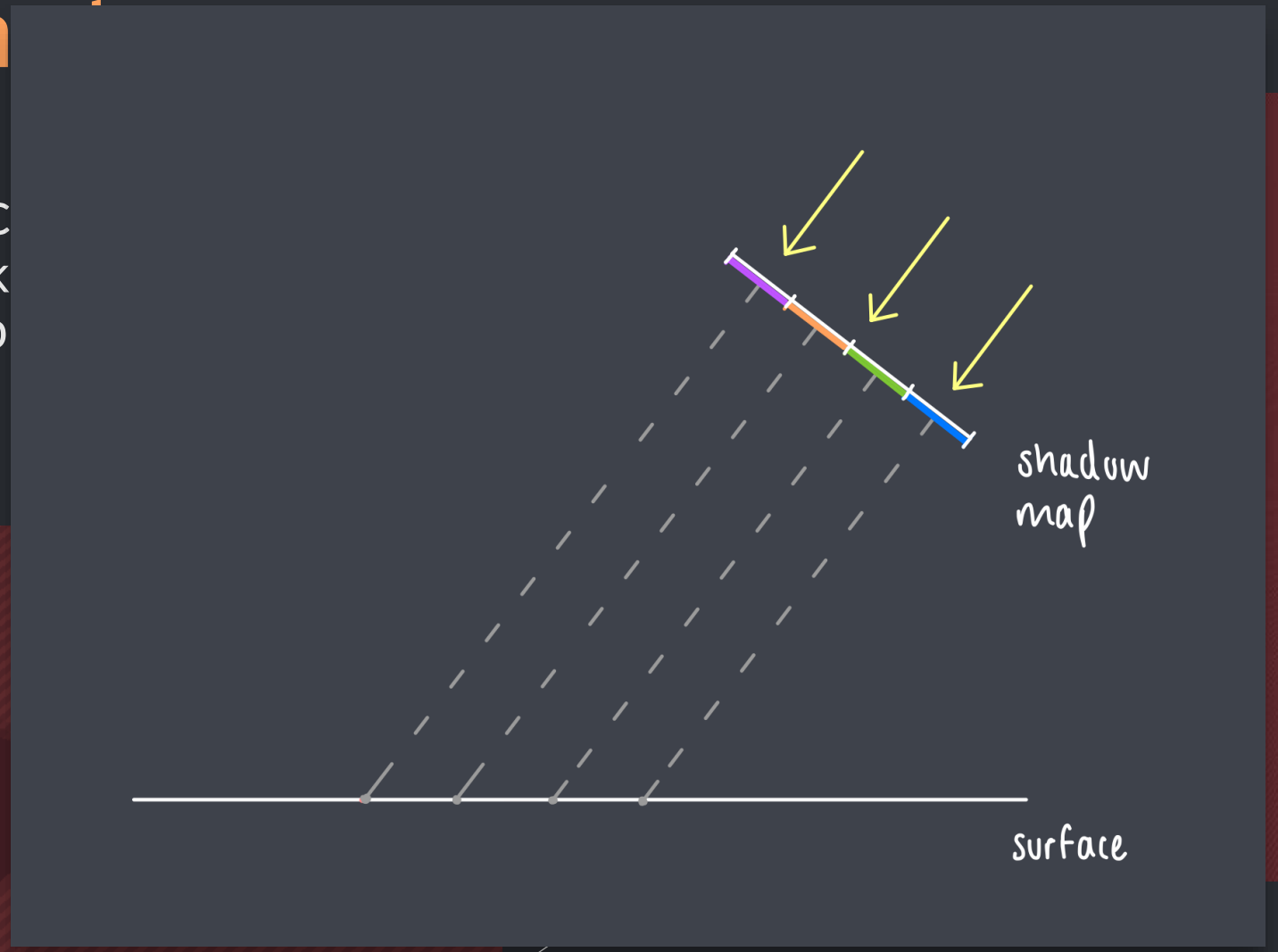
Artefacts

- Shadow Acne
  - Moiré-like pattern
  - Sol^n: apply small bias



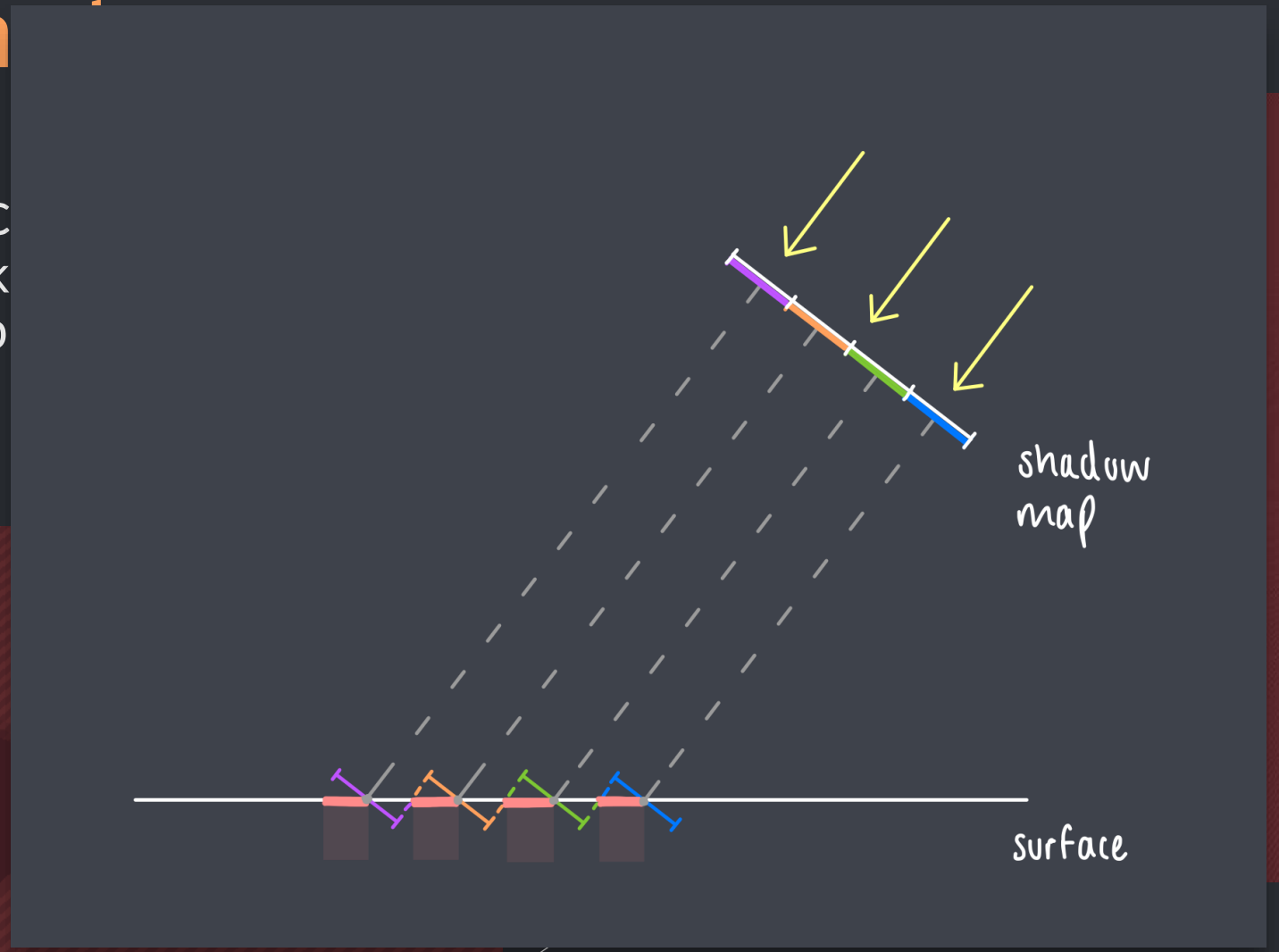
Artefa

- Shadow Ac
  - Moiré-lik
  - Sol^n: app



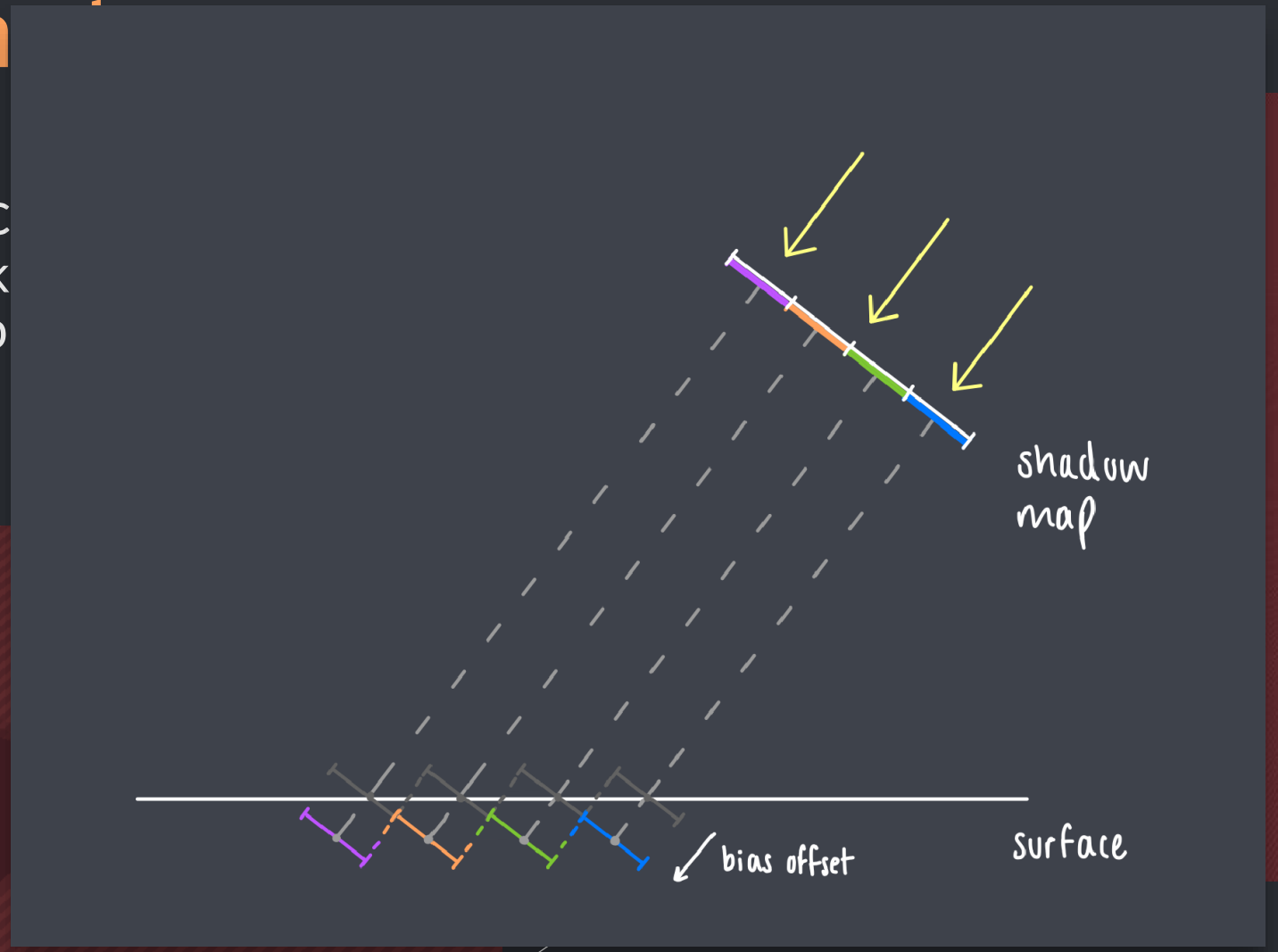
Artefa

- Shadow Ac
  - Moiré-lik
  - Sol^n: app



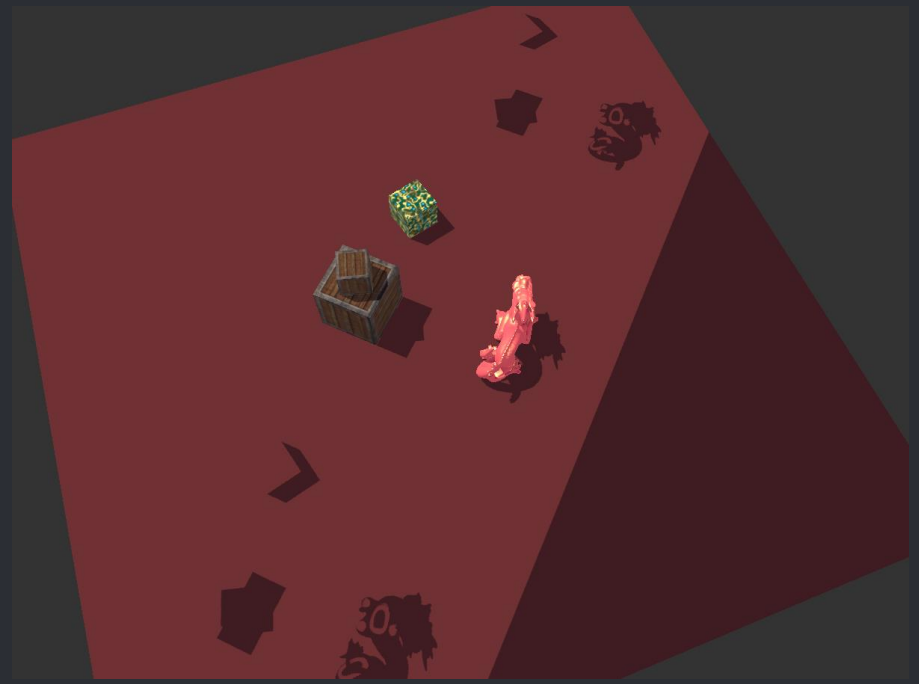
Artefa

- Shadow Ac
  - Moiré-lik
  - Sol^n: app



Artefacts

- Shadow Acne
  - Moiré-like pattern
  - Sol^n: apply small bias
- Oversampling Issues
  - Points outside depth map appear in shadow
  - Sol^n:
    - texture parameters
    - force fragments to be lit when $z_f > 1.0$



Artefacts

- Shadow Acne
  - Moiré-like pattern
  - Sol^n: apply small bias
- Oversampling Issues
  - Points outside depth map appear in shadow
  - Sol^n:
    - texture parameters
    - force fragments to be lit when $z_f > 1.0$
- Jagged Edges
  - Jagged, blocky edges to shadows
  - Sol^n: Percentage Close Filtering (PCF)

Demo time! :)

Shadow Mapping In Context

- A Several advantages:
  - Fast on modern GPUs
  - Relatively easy to implement
  - Transparent Shadows
- …and some drawbacks:
  - Aliasing!

    …but many + advanced shadow mapping techniques improve this (at the expense of resources or flexibility)

    e.g. Cascaded Shadow Maps, Percentage Closer Soft Shadows…
  - Omni-directional Shadow Mapping (for point lights) requires + renders

Summary – key takeaways

- A technique for rendering shadows in real-time 3D graphics
  - Shadows add realism, convey depth & convey spatial relationships between objects
- Main foundation technique nowadays
  - Advanced techniques provide more accurate results
  - Many new ones being developed
- Try to implement it!

Q&A – any questions?