Modular Graphics Pipelines

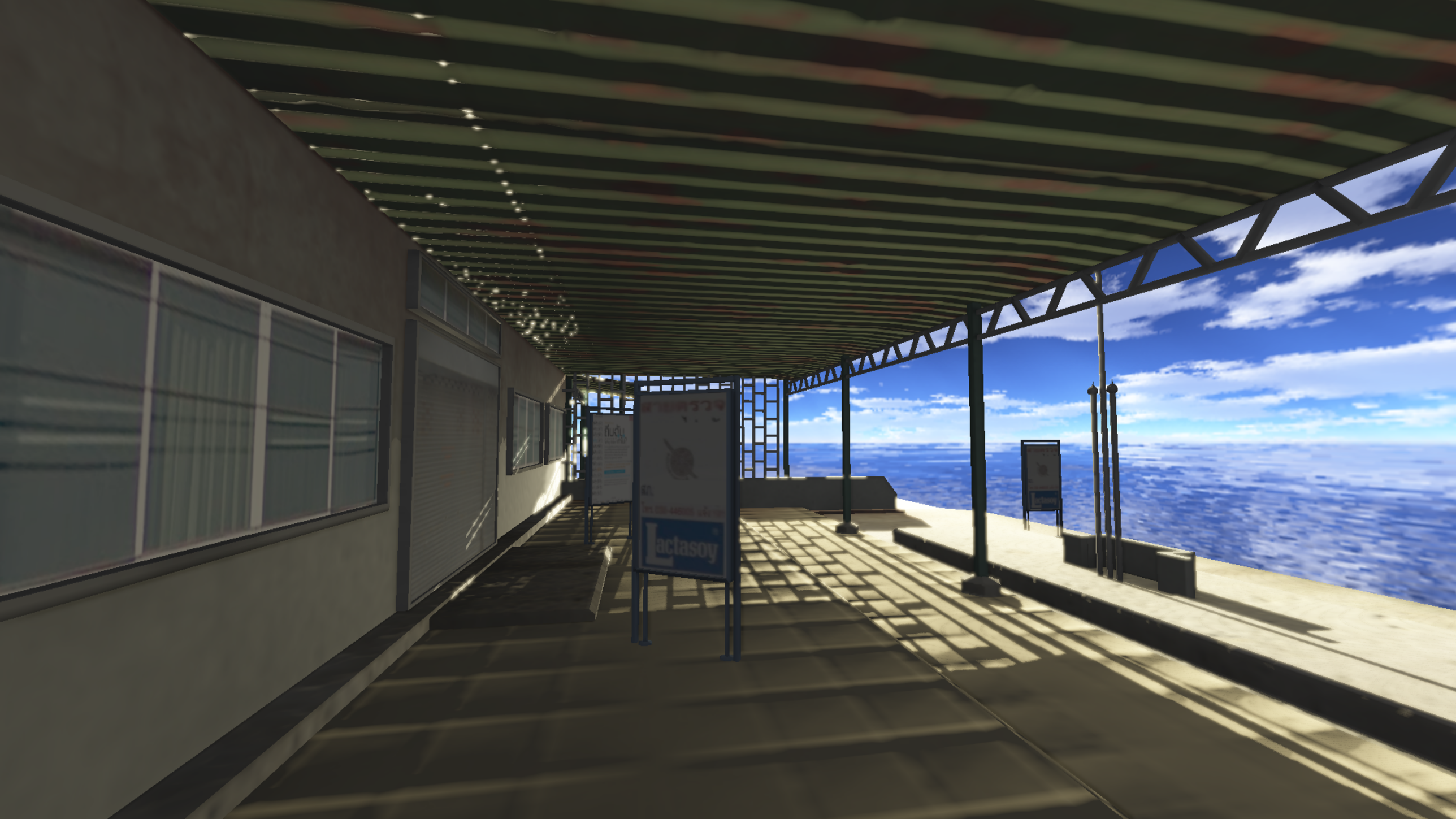
https://shinmera.com

Talk Overview

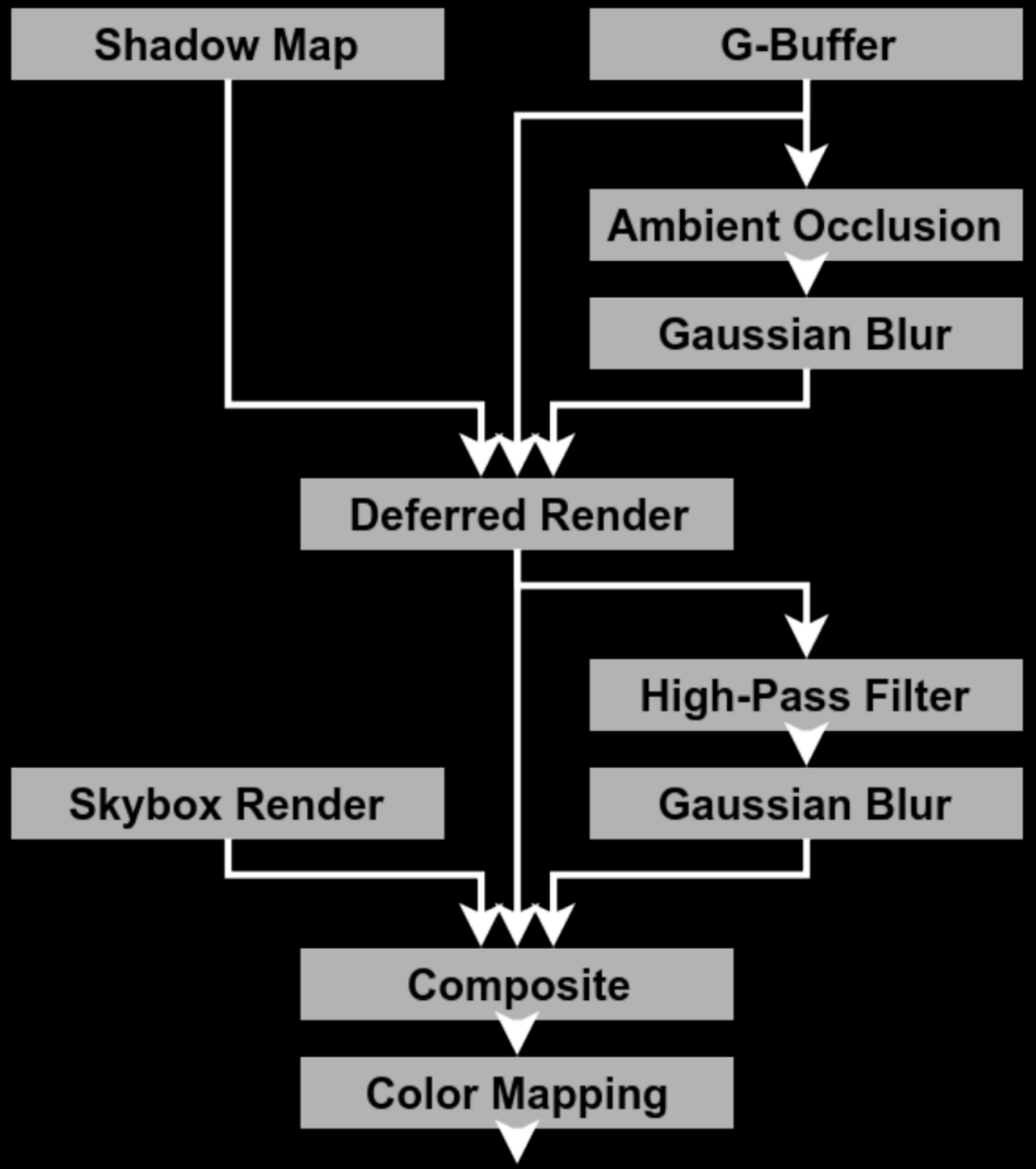
- 0. Terminology
- 1. A sample scene
- 2. How it's put together
- 3. How the paper helps
- 4. What it fails at

Terminology

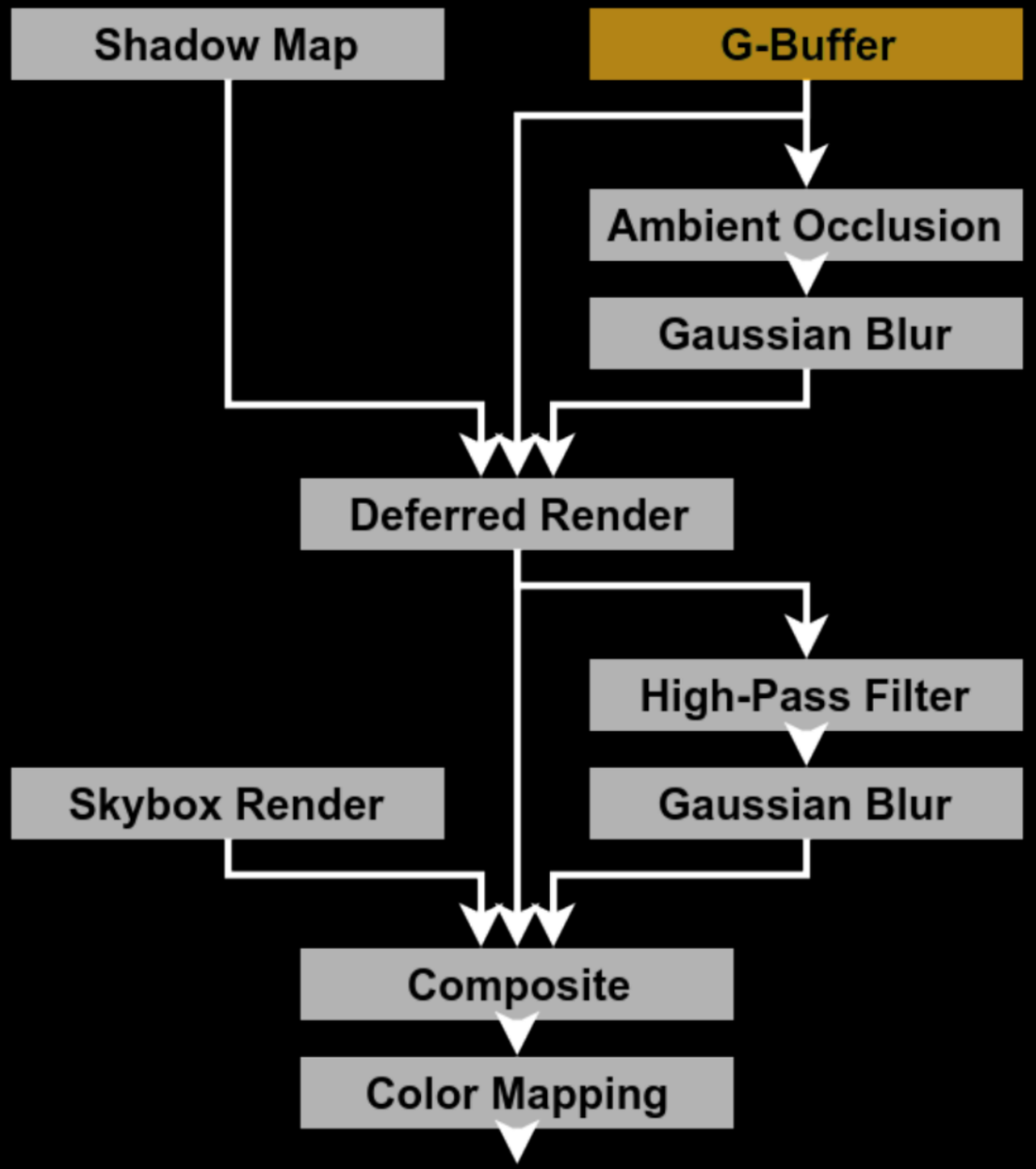
- Shader: code run on the GPU for rendering
- Texture: an image buffer on the GPU
- Pass: one or more rendering steps
- Pipeline: a series of connected passes

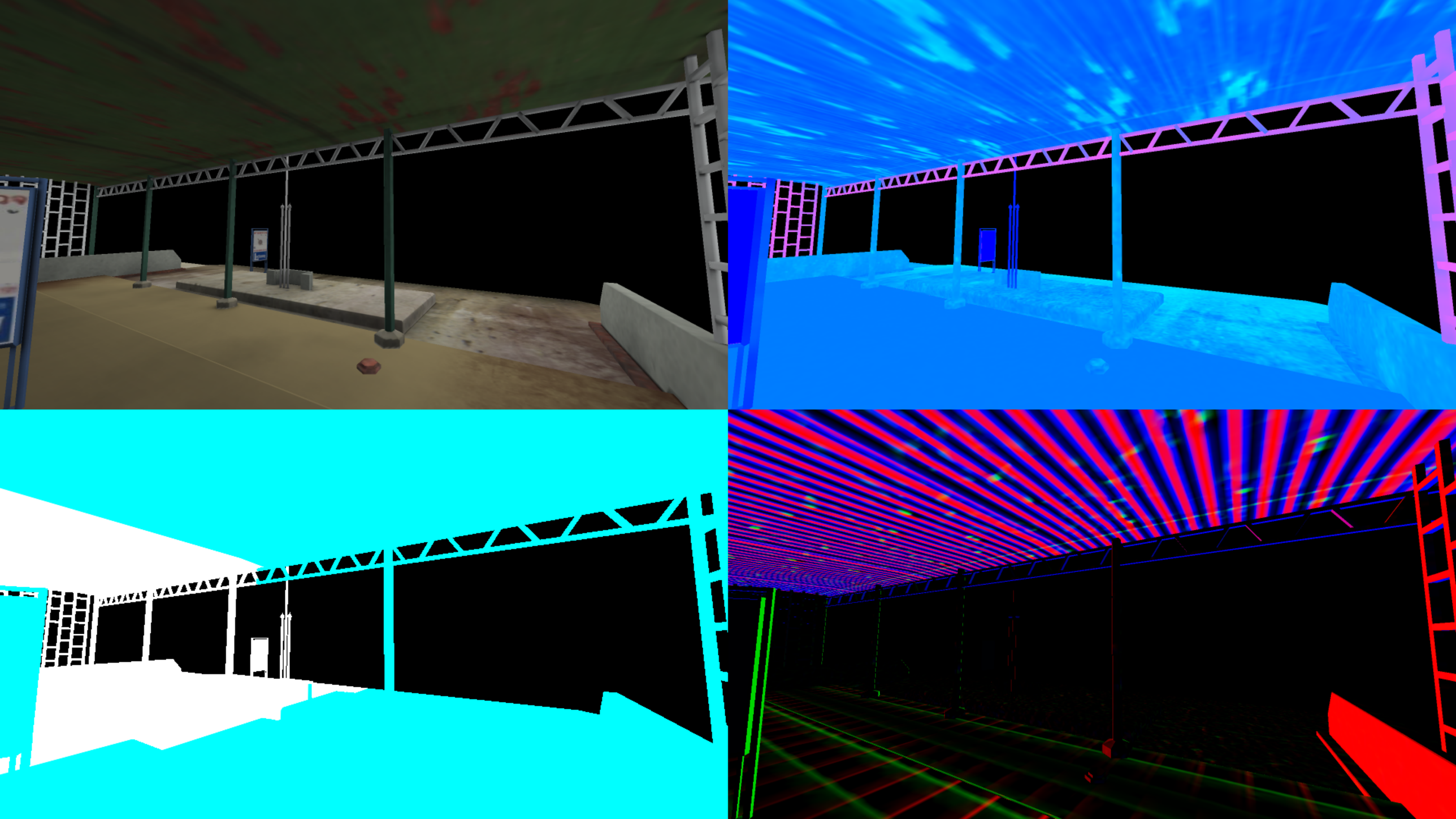

Lactasoy

For such a scene we need:

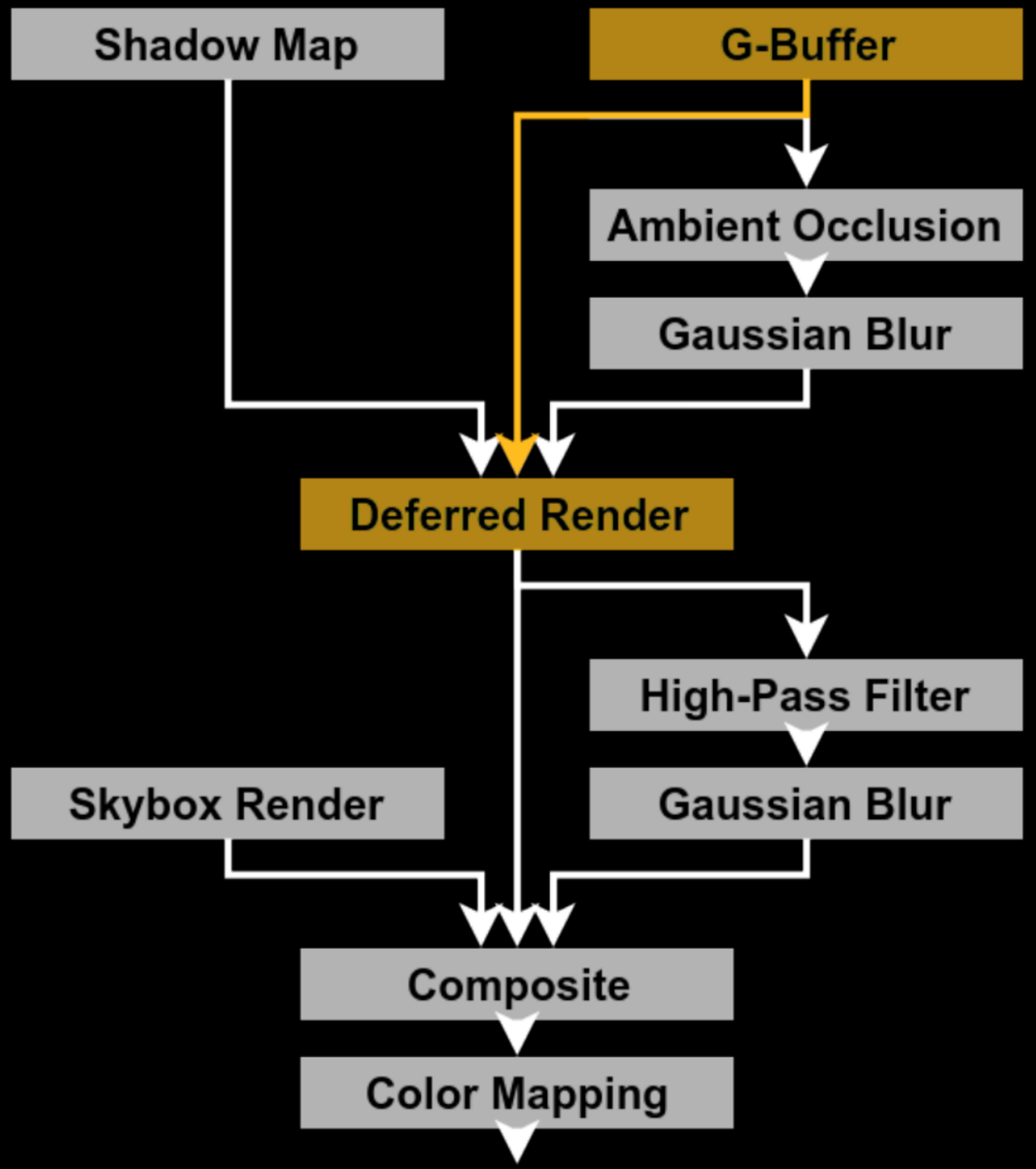


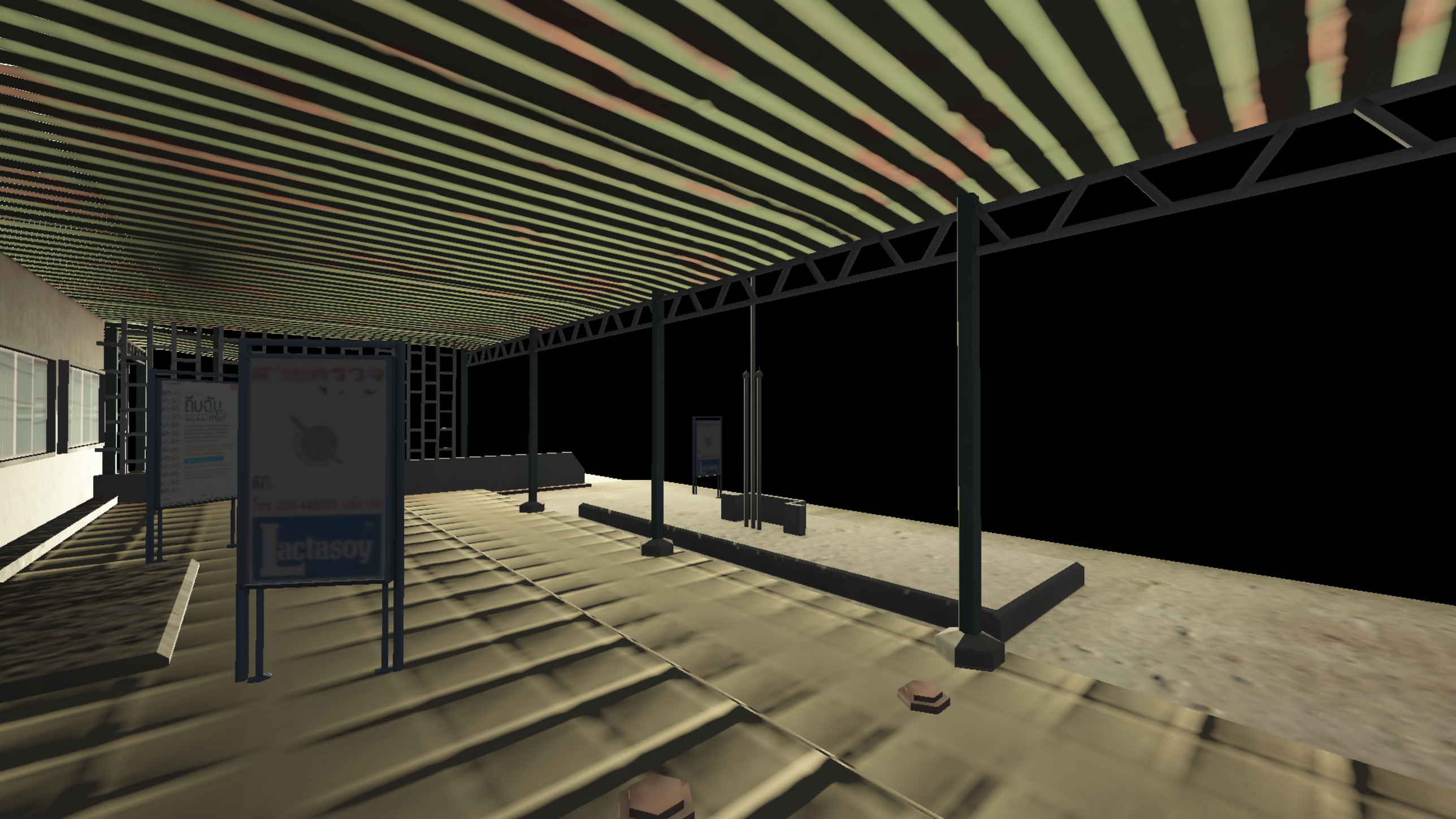
Overview


Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping

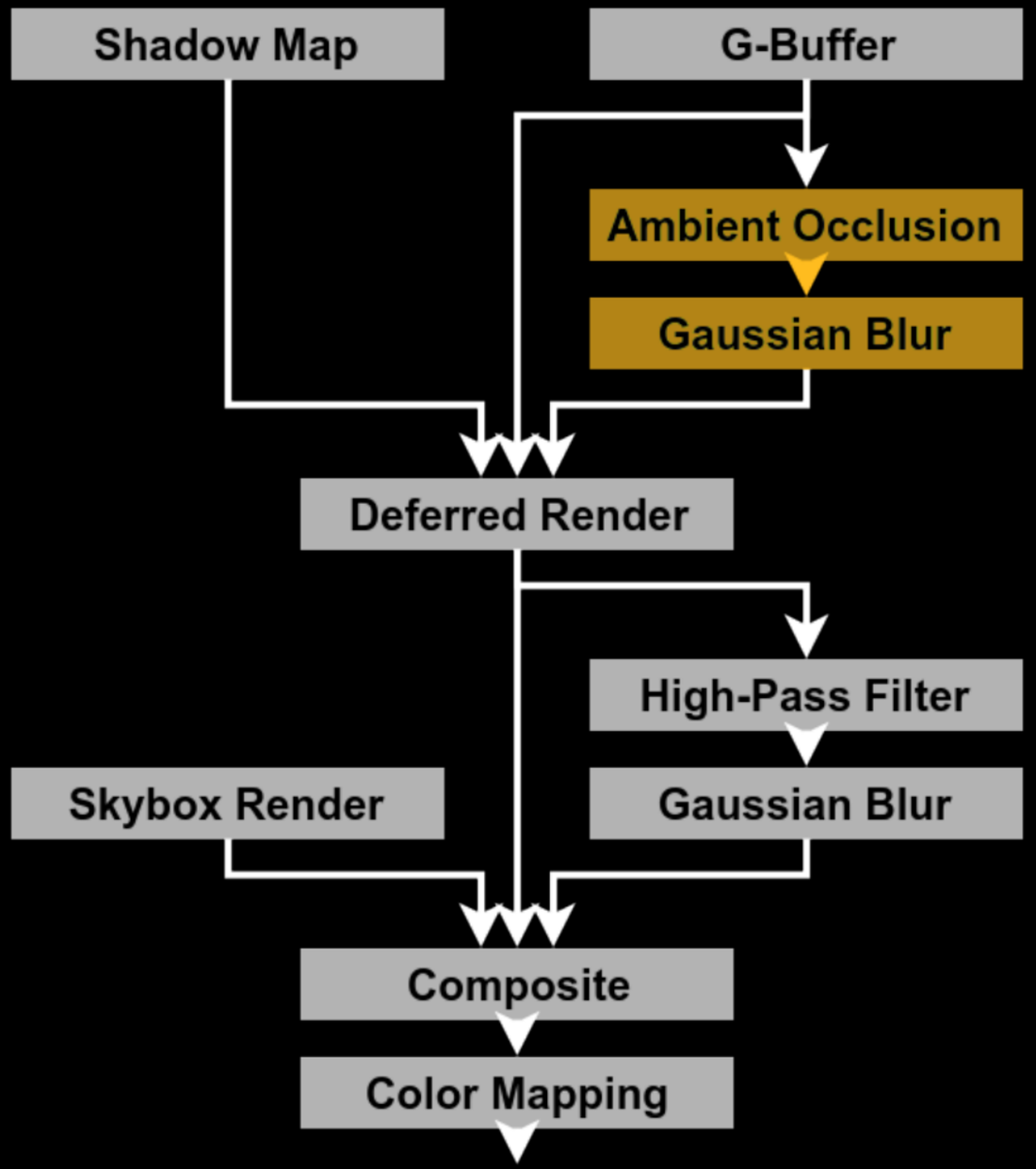


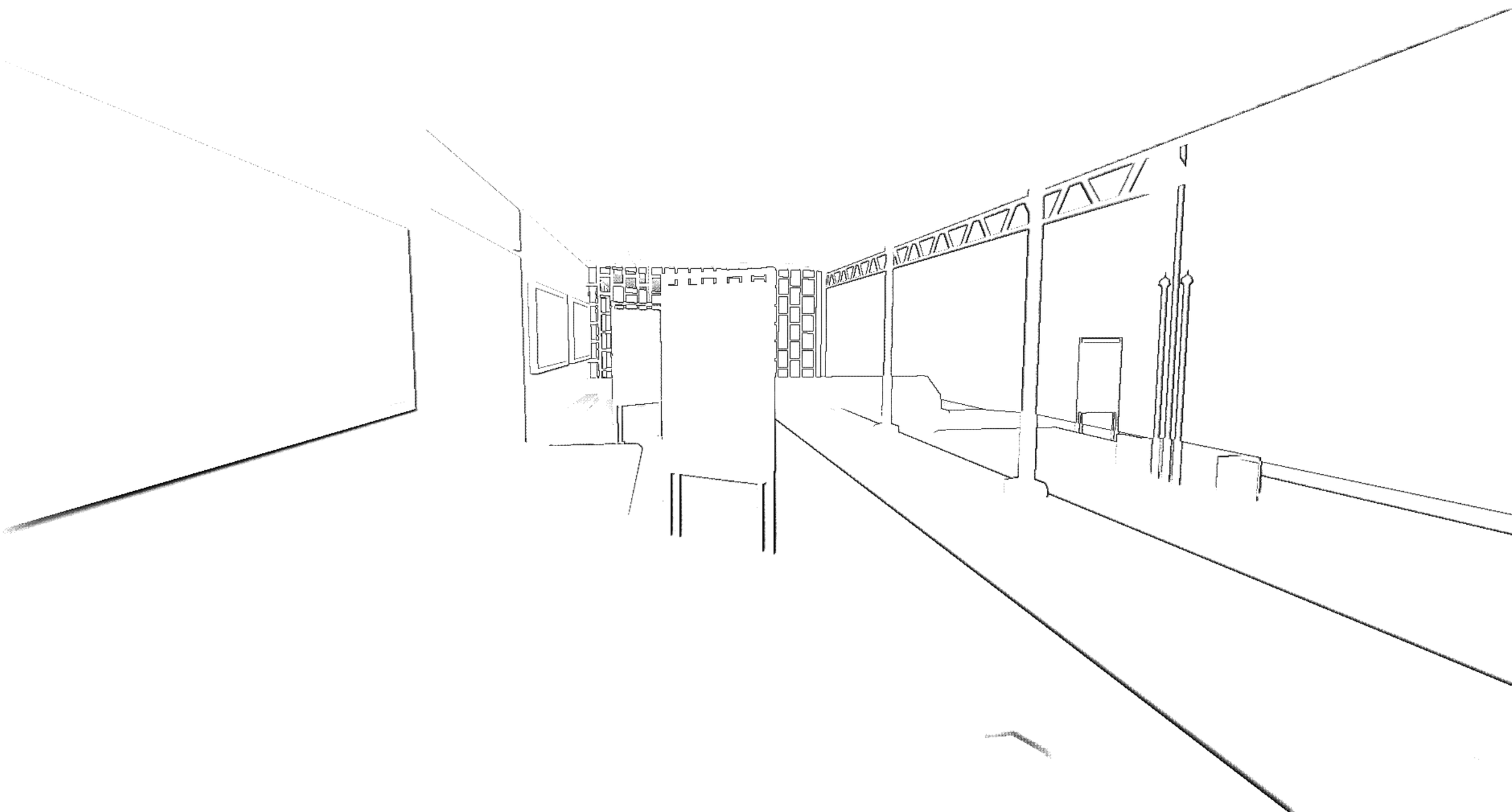
Overview


Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping

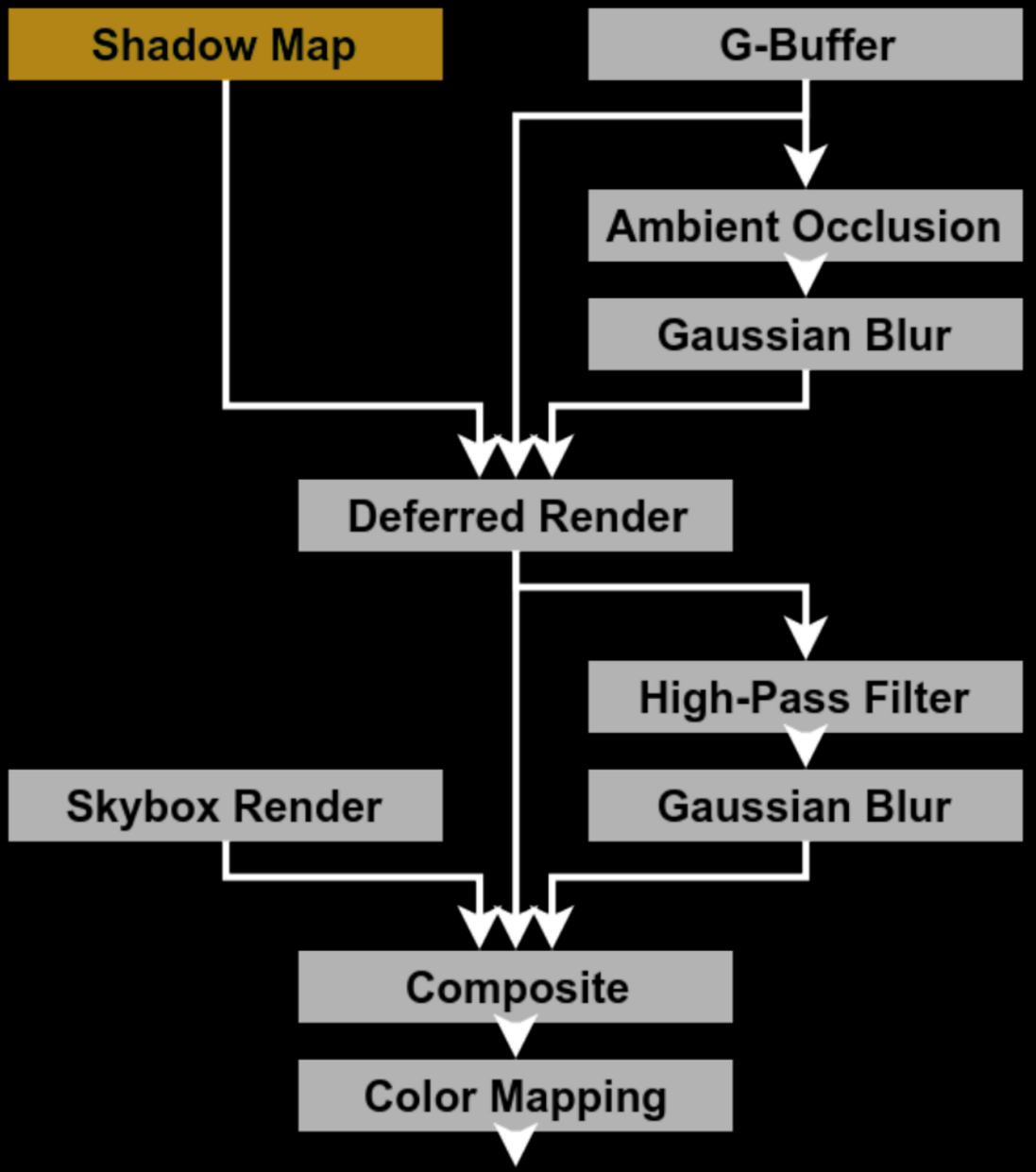

ถึงฉัน
Why Kids ทำไม?
Lactasoy

Overview

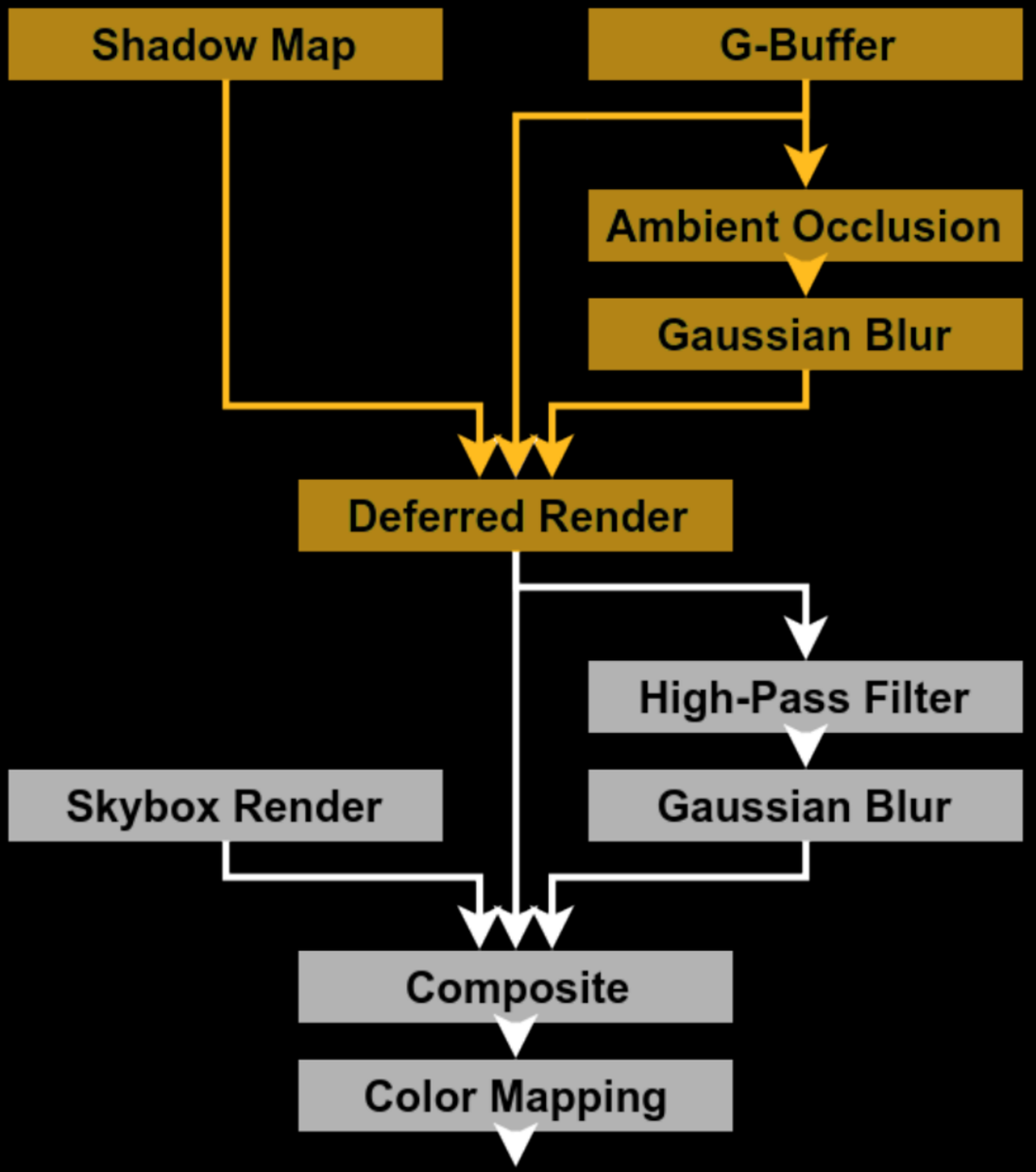

Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping

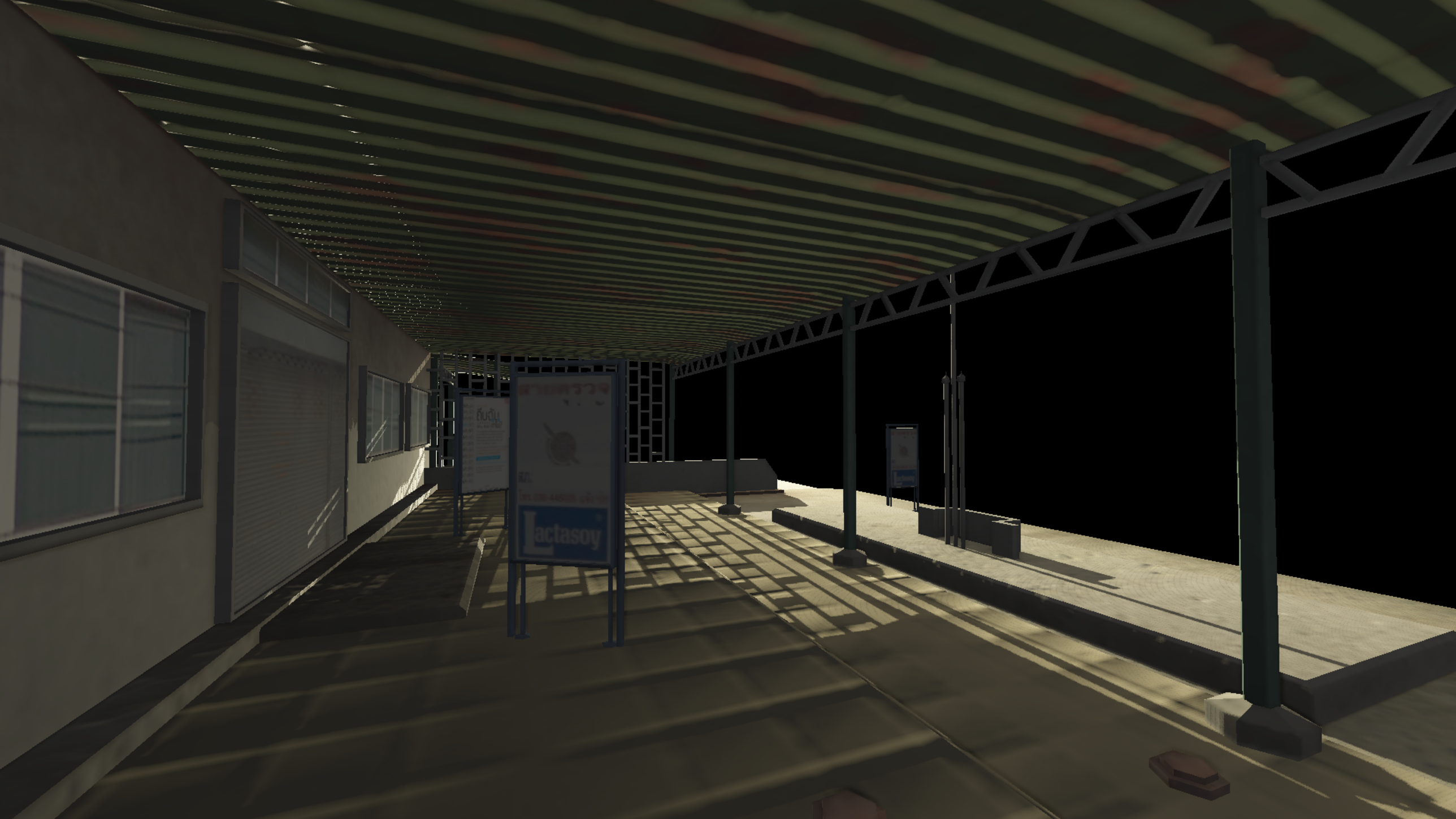


Overview

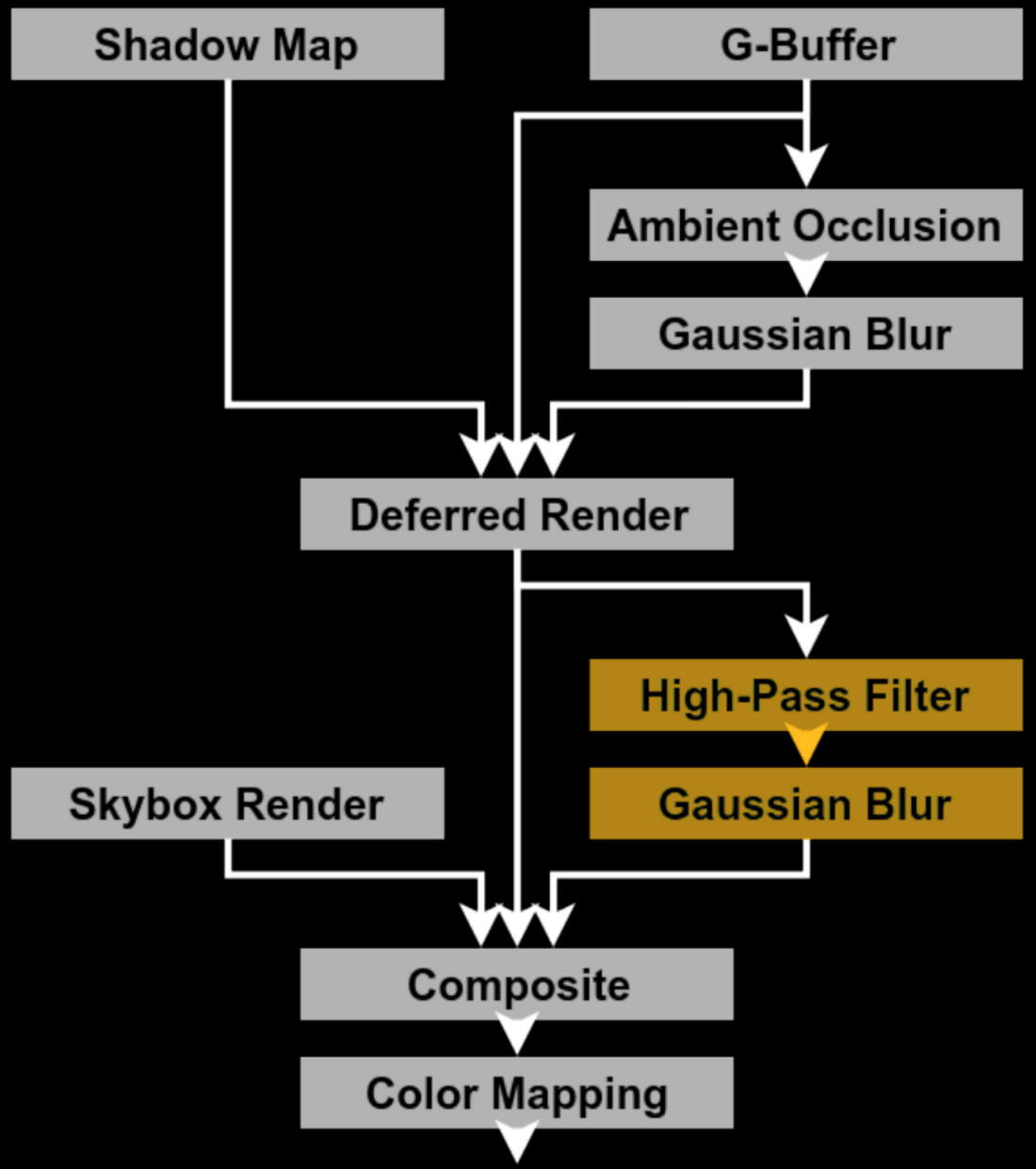

Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping

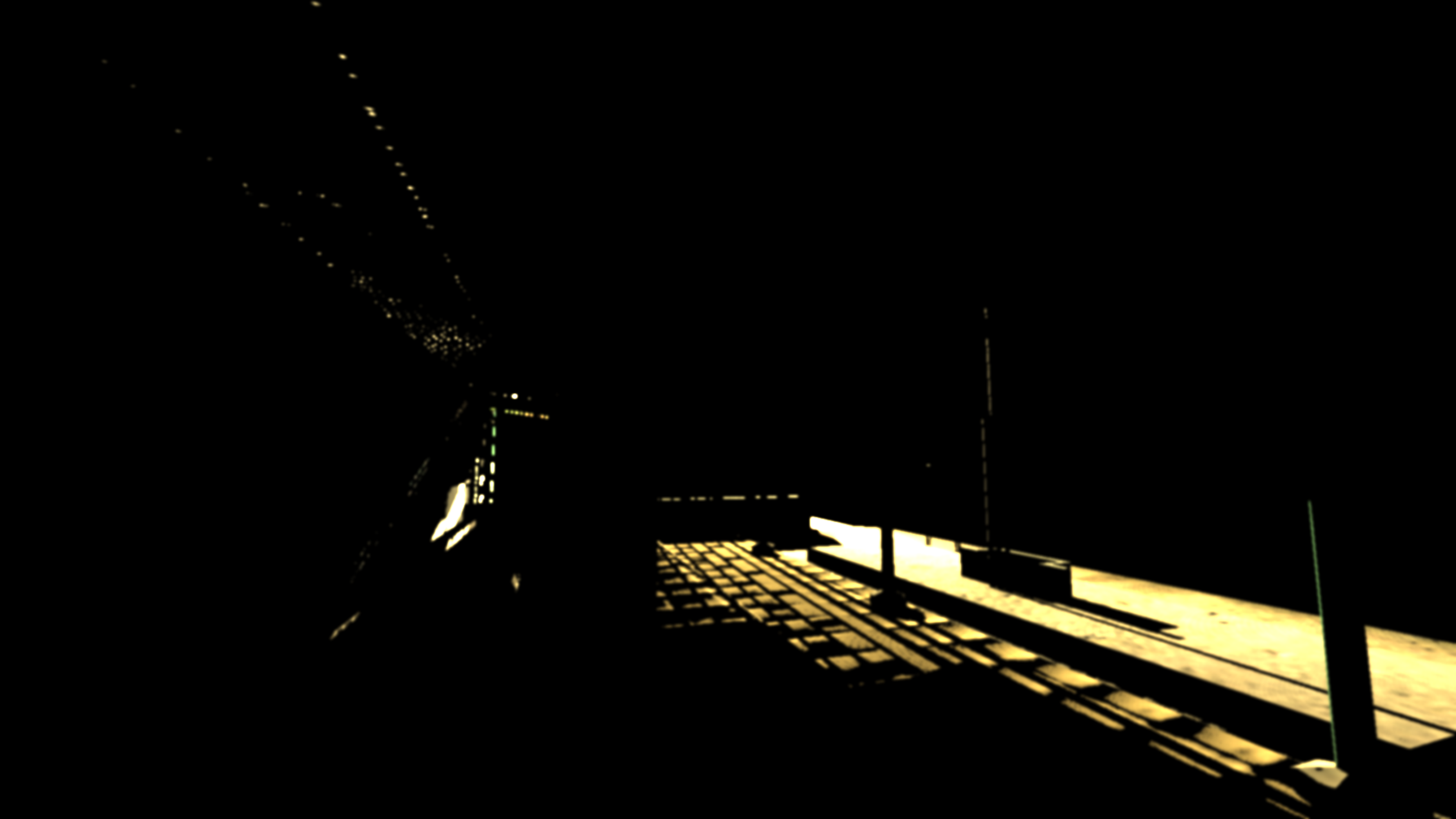
Overview


Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping

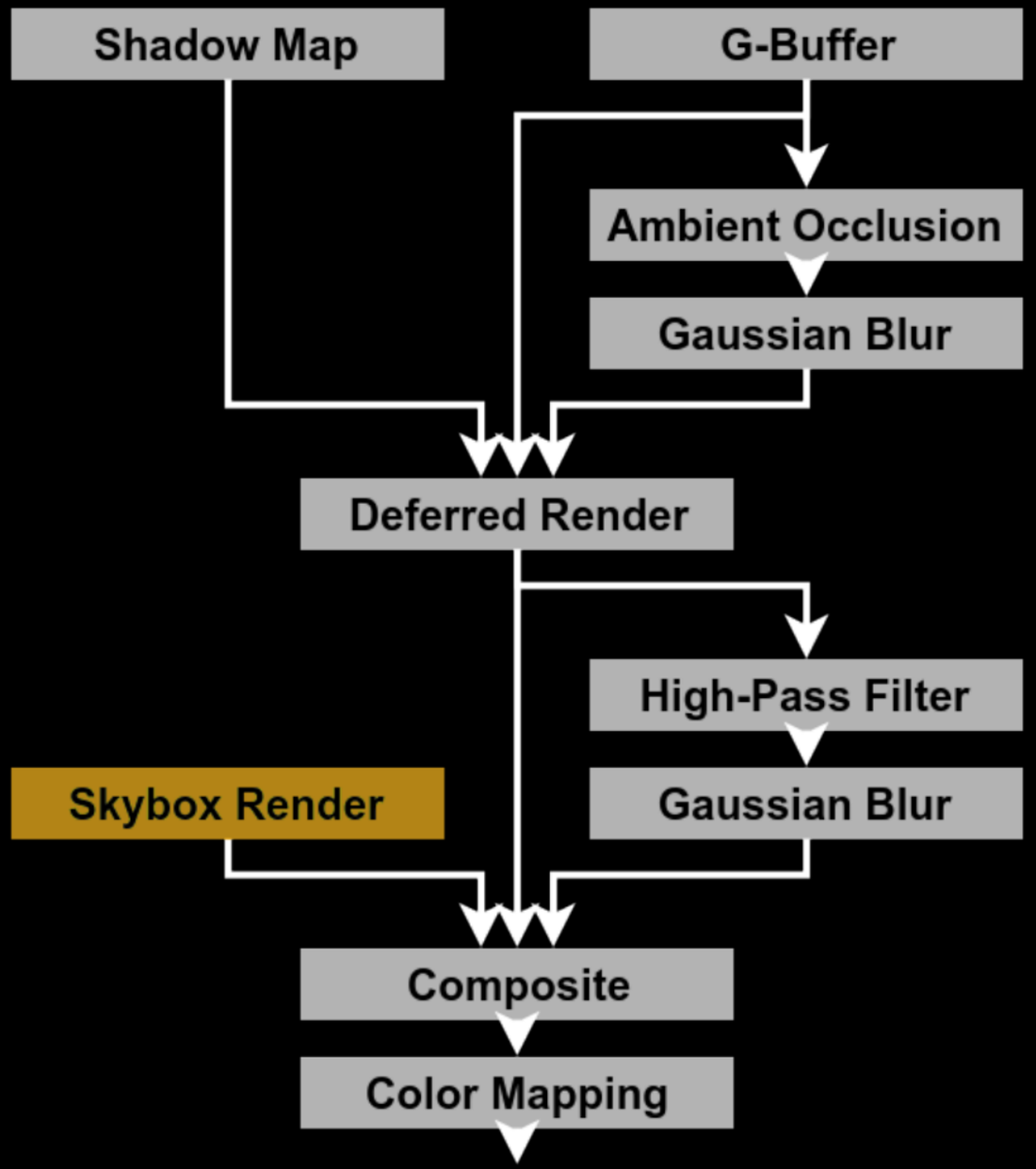

Lactasoy

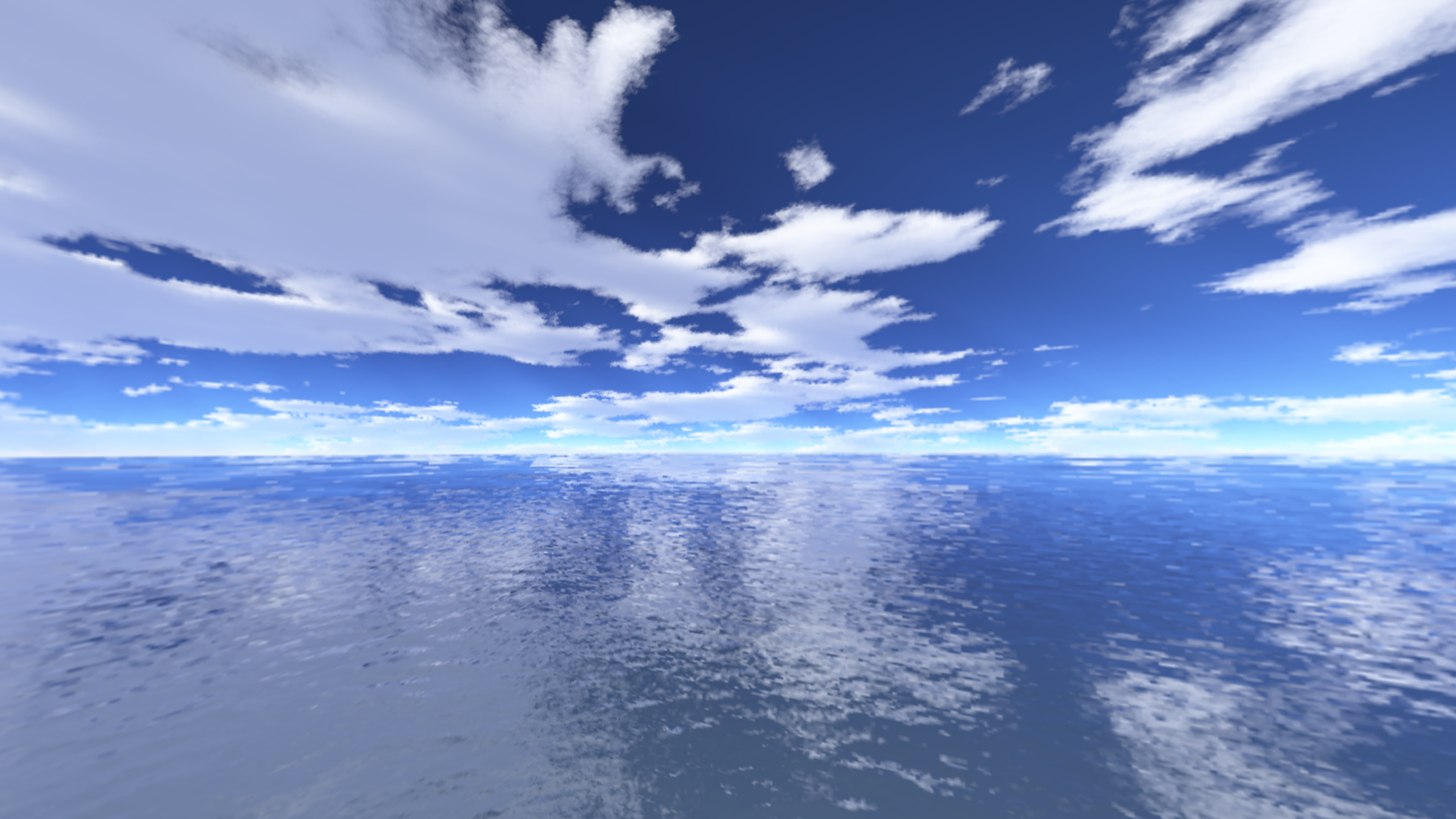
Overview

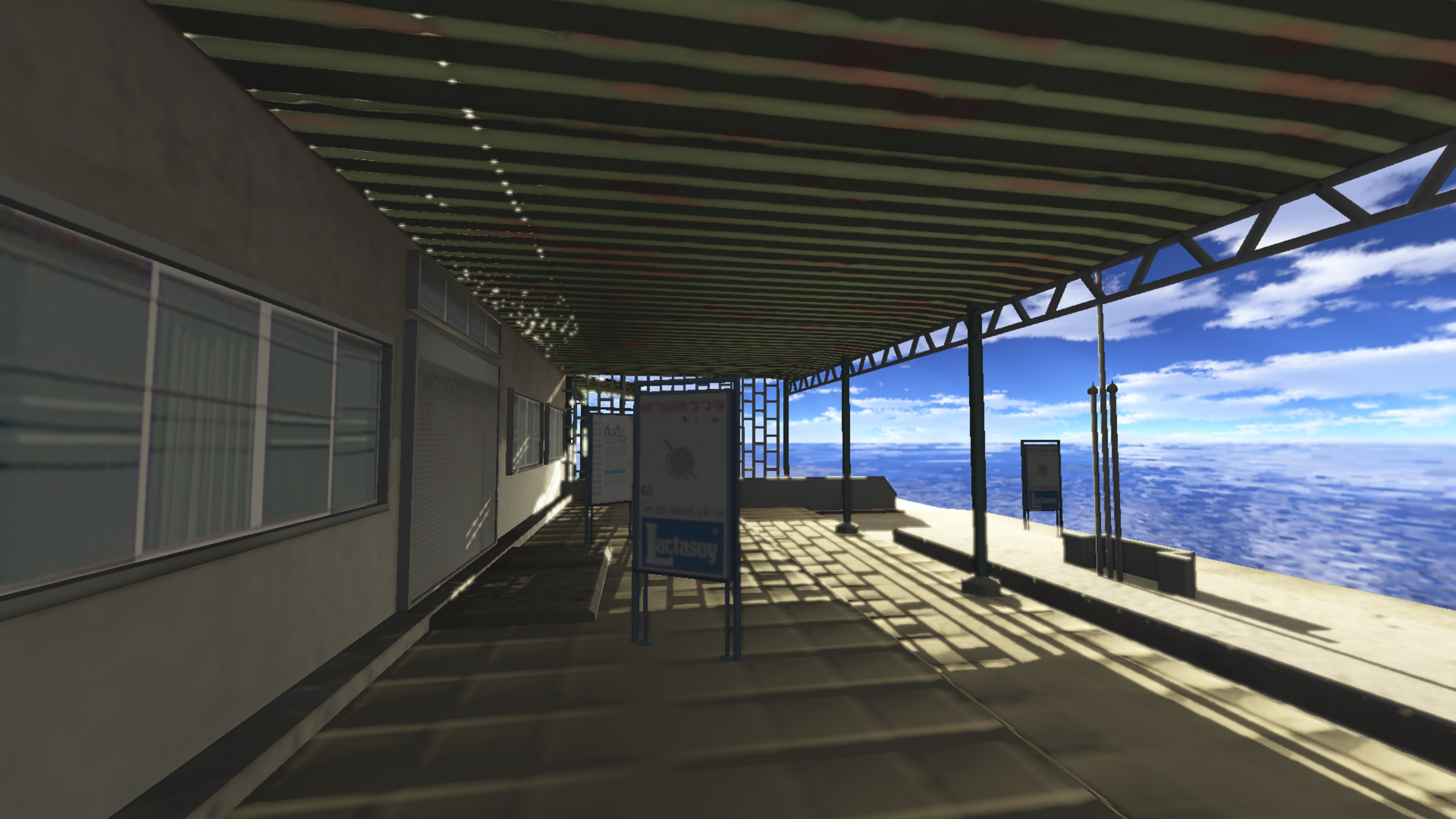

Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping



Overview


Shadow Map
G-Buffer
Ambient Occlusion
Gaussian Blur
Deferred Render
High-Pass Filter
Skybox Render
Gaussian Blur
Composite
Color Mapping




Lactasoy

Architectural Challenges

- Pipelines become complex
- Lots of buffer state to manage
- Many parts interact tightly

Abstract Pipelines

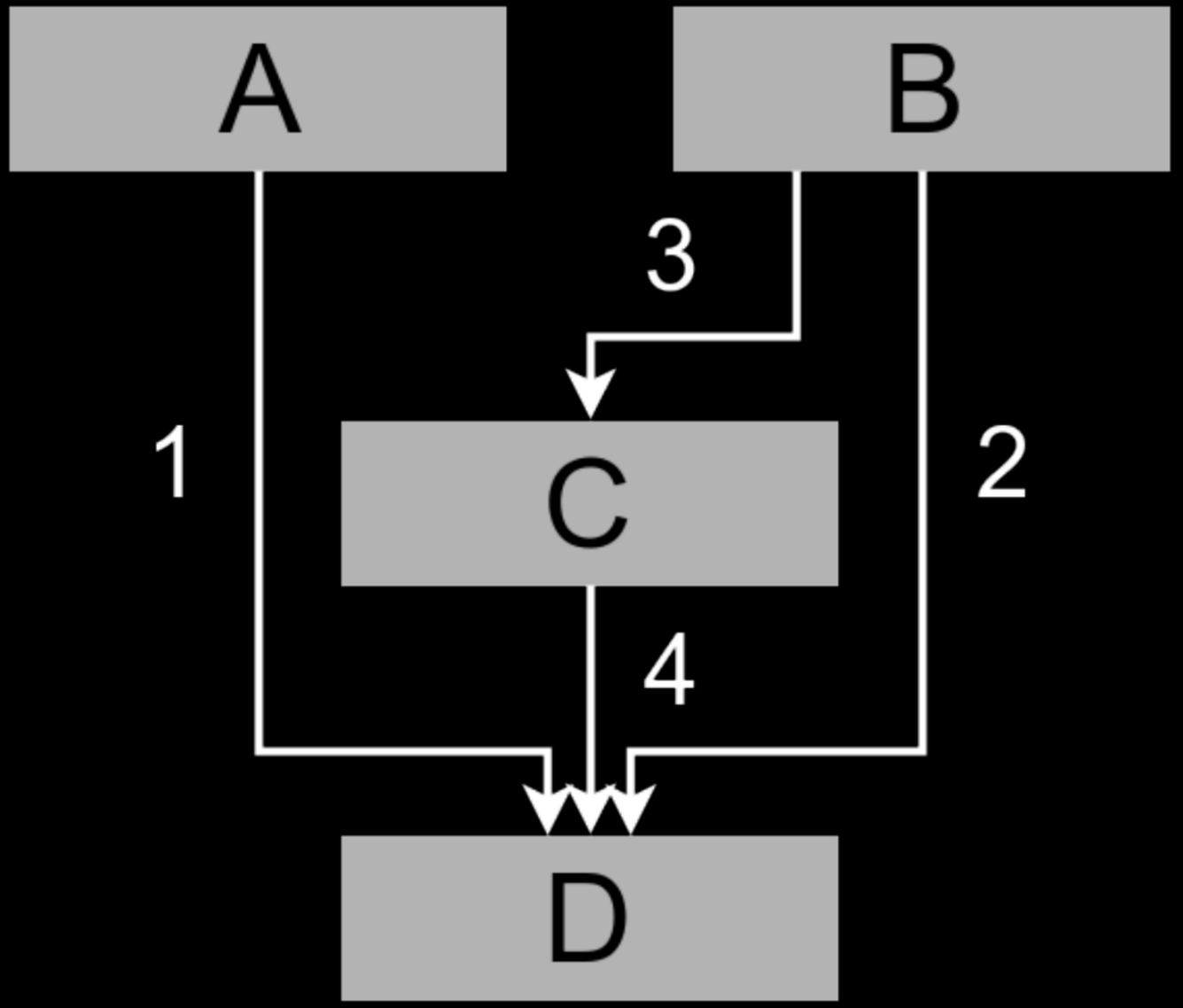
Create a new pass class

```
(define-shader-pass ssao-pass ()
  ((depth      :port-type input
               :texspec (:internal-format :depth-component))
   (normal     :port-type input)
   (occlusion  :port-type output
               :texspec (:internal-format :red))))
```

Connect passes in the pipeline

```
(let ((pipeline (make-instance 'pipeline))
      (g-buffer (make-instance 'g-buffer-pass))
      (ssao-pass (make-instance 'ssao-pass)))
  (connect (port g-buffer 'depth) (port ssao-pass 'depth))
  (connect (port g-buffer 'normal) (port ssao-pass 'normal)))
```

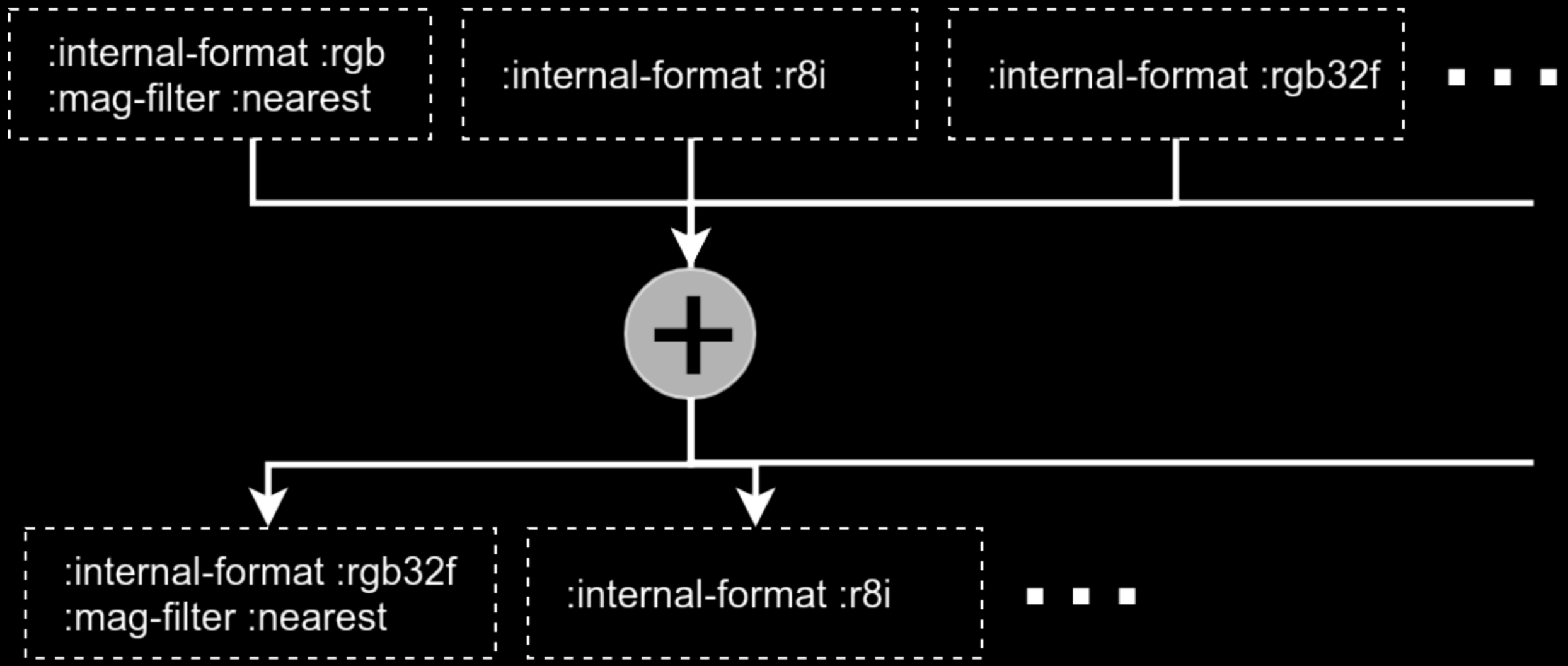
Data Flow Restrictions



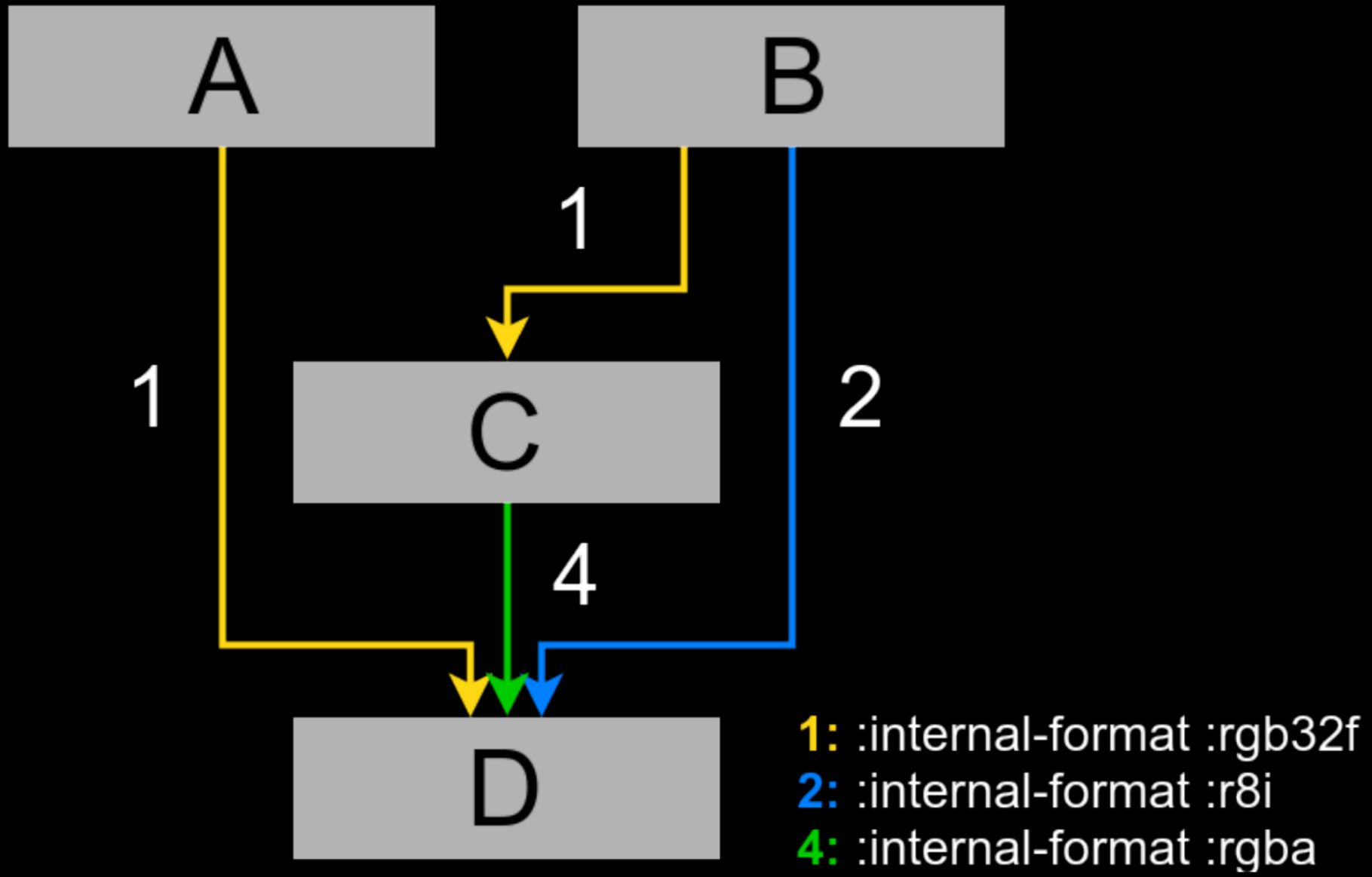
Texture State

- Texture Size
- Channel count
- Channel precision
- Channel data type
- Texture usage
- Interpolation filter
- Mipmapping levels, lod, and bias
- UV addressing mode and border color
- Texture data storage mode
- Anisotropy
- Multisampling

Texture Specification Joining



Automated Buffer Allocation



Modular Shader Composition

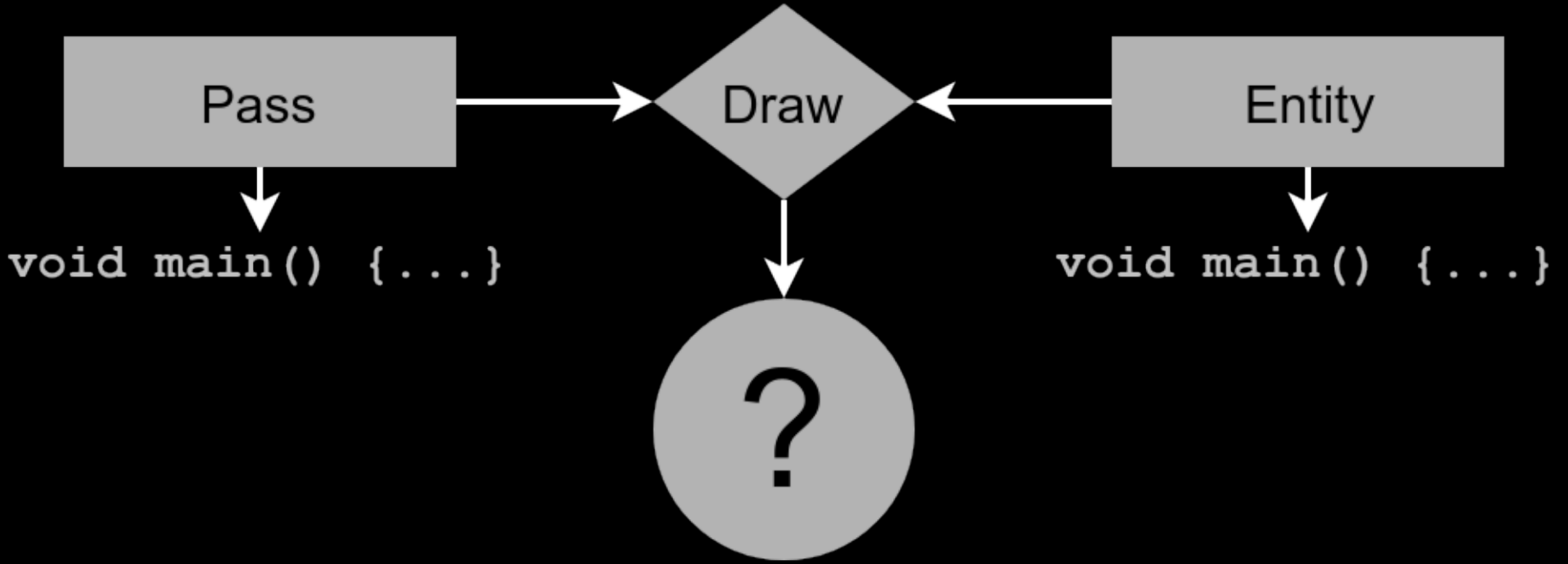
Define shader classes by inheritance

```
(define-shader-pass renderer (high-color-pass
                              hdr-output-pass
                              deferred-render-pass
                              shadow-render-pass
                              ssao-render-pass)
  ())
```

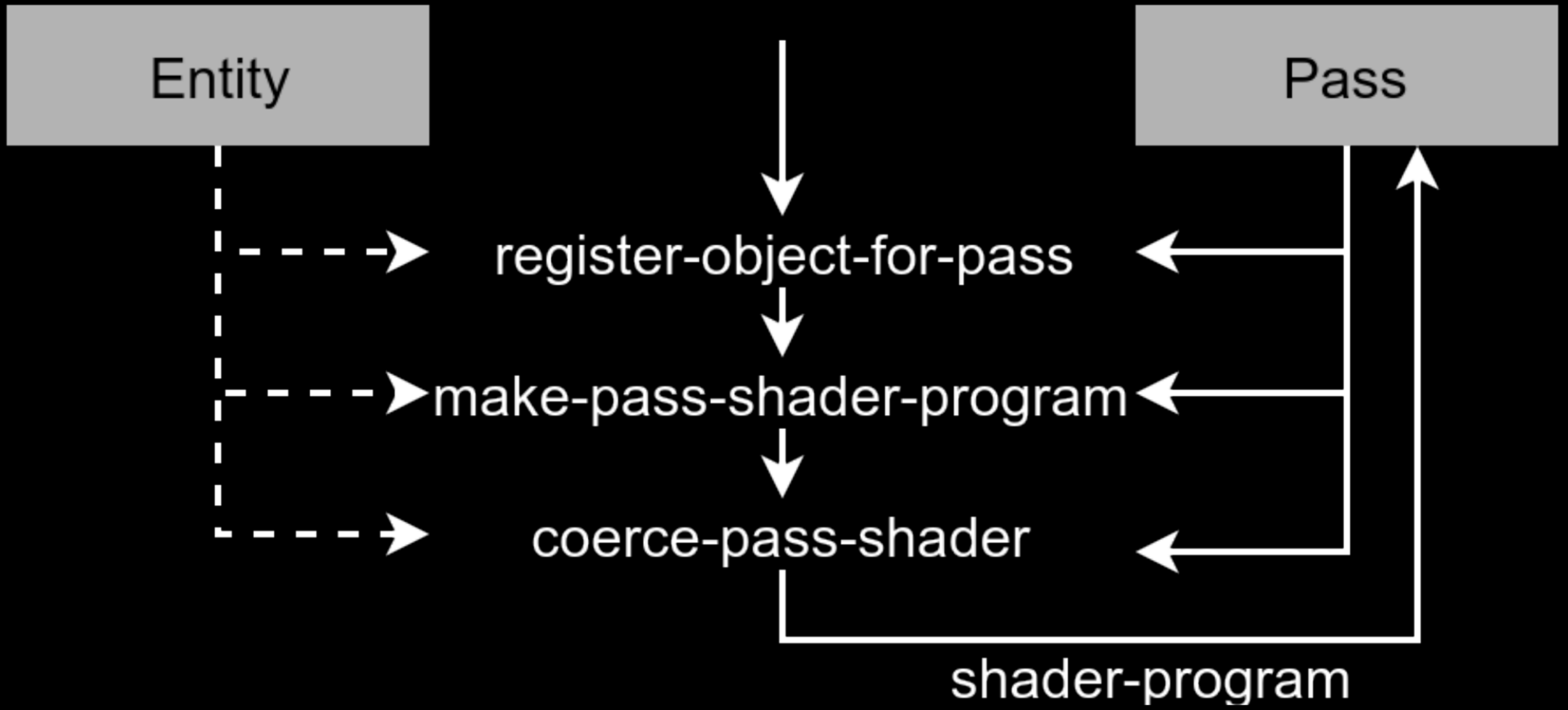
Associate shader code to classes.

```
(define-class-shader (renderer :fragment-shader)
  "void main(){
  primary_strength = 1-shadow_strength();
  ambient_strength = occlusion_strength();
}")
```

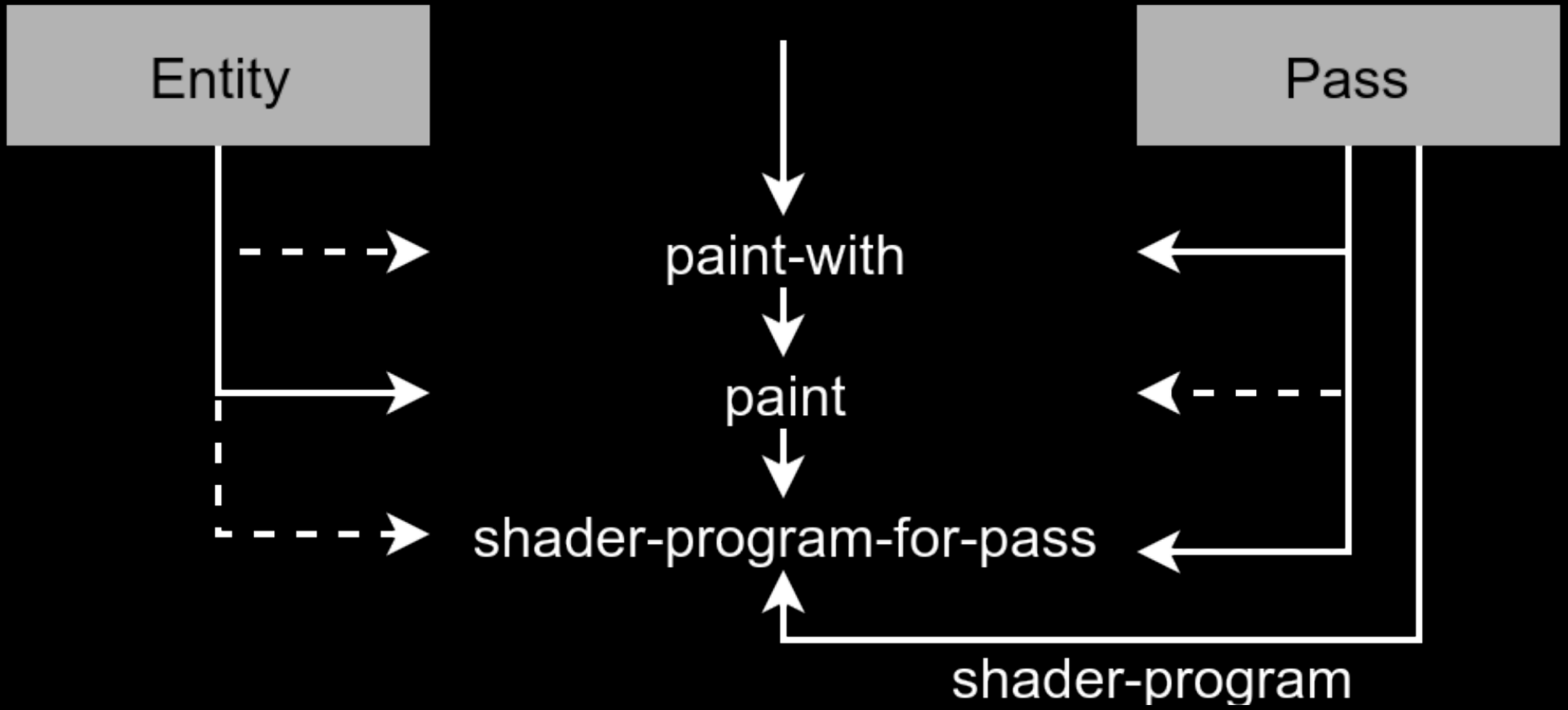
Combining Passes and Objects



Combining Passes and Objects



Combining Passes and Objects



Issues

- Code analysis very primitive
- Cannot capture user intent
- Need to anticipate combination

Future Work

- Code inference and analysis
- Use-relation tracking
- Shader I/O interface abstraction
- Dynamic pipelines