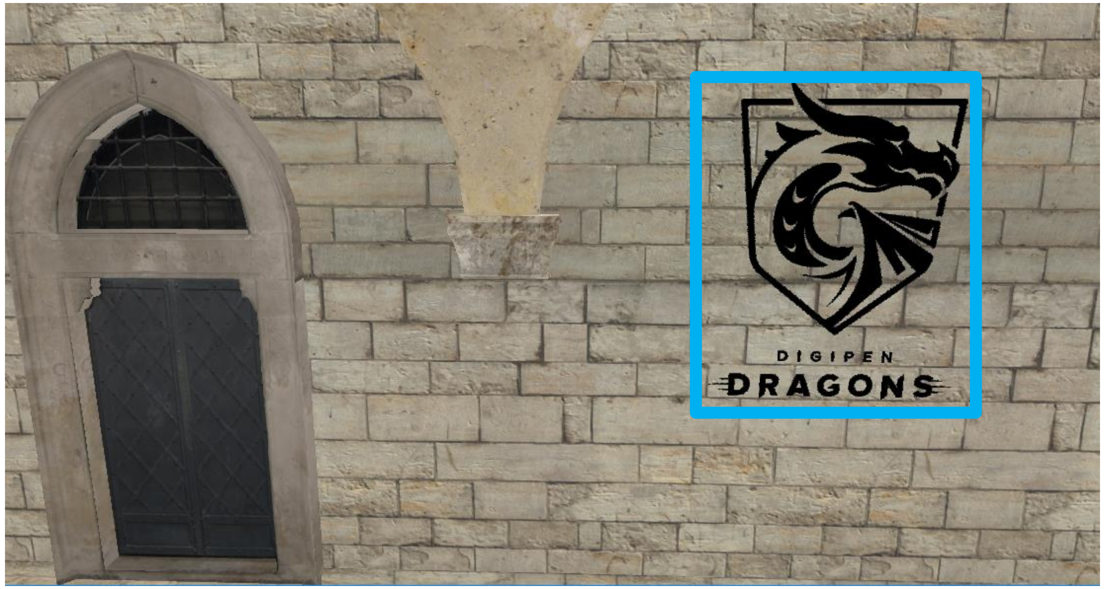
Decals

Decals

- What is a decal?
  - Pattern or image that can be moved to another surface



Decals

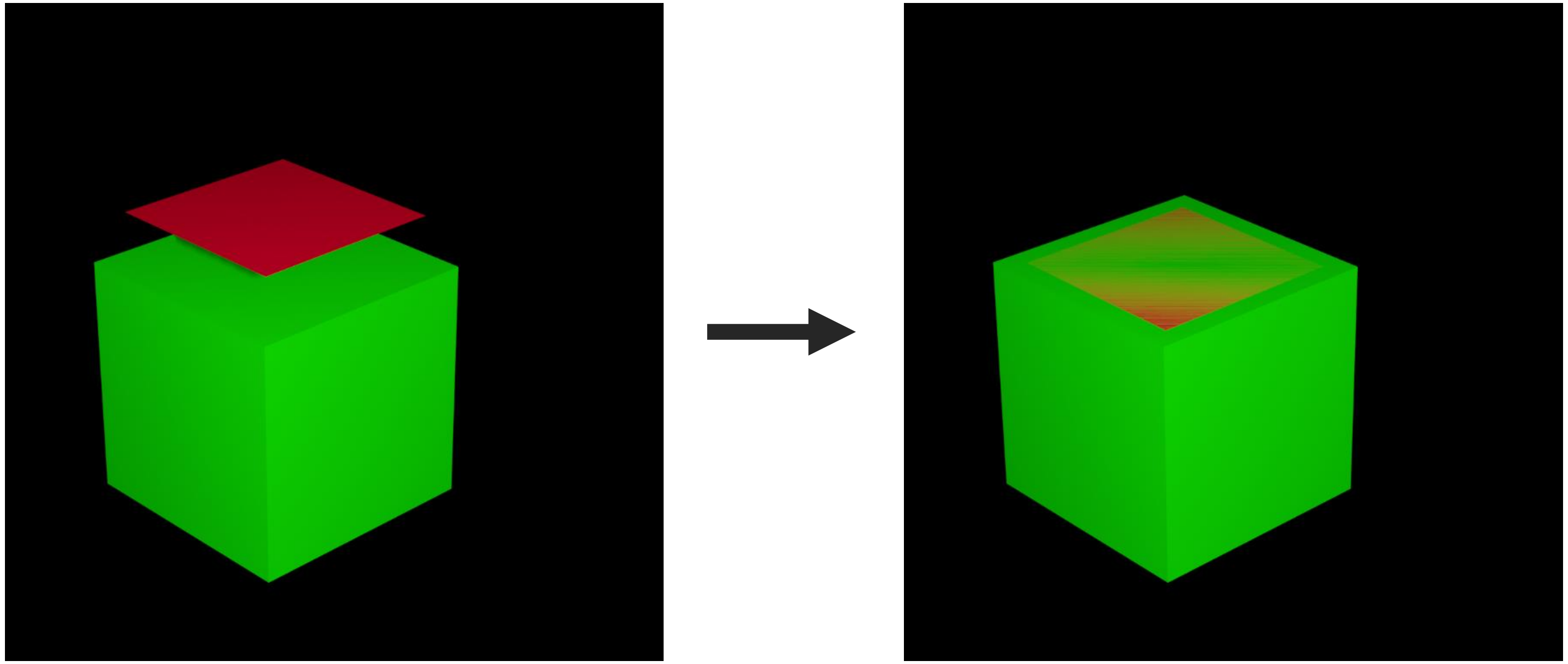
- Why do we need them?
  - To enhance the scene dynamically or statically
  - Dynamic
    - Bullet holes
    - Painted floor/walls
    - Explosion marks
    - …
  - Static
    - Cracks
    - Wall imperfections
    - Signs
    - …

Decals

- Just a texture on top of a mesh…
- Sounds simple and it actually is…
- How can we add them?

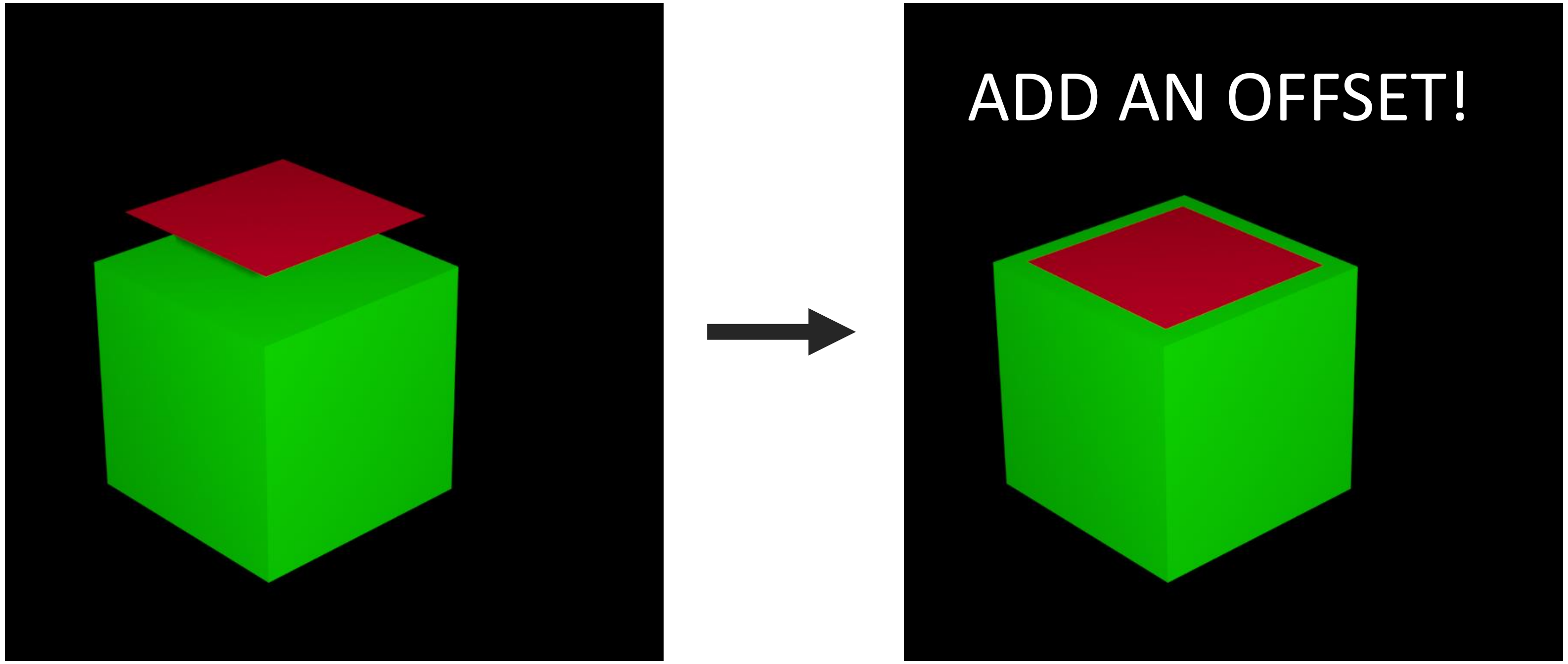
Decals: Approach 1

- Project the quad onto the geometry
- Create mesh and render separately with the texture



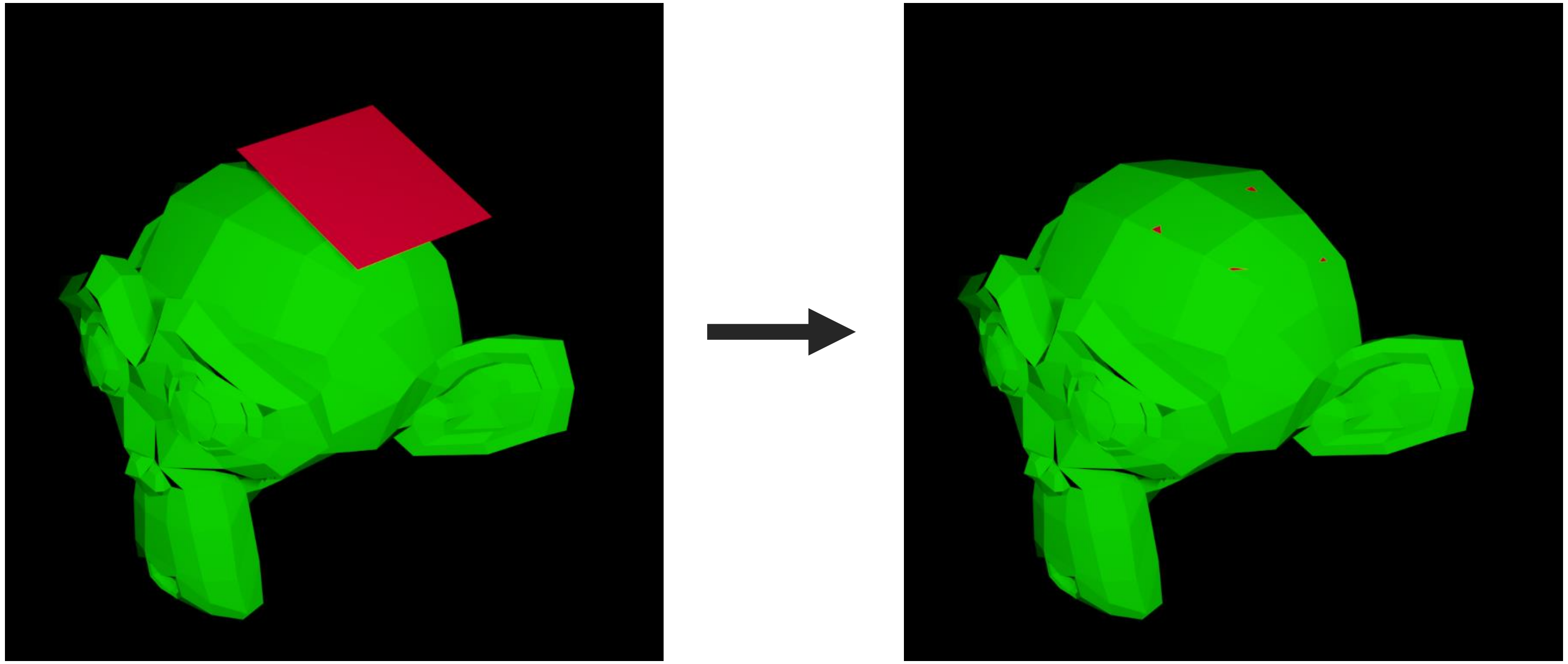
Decals: Approach 1

- Project the quad onto the geometry
- Create mesh and render separately with the texture



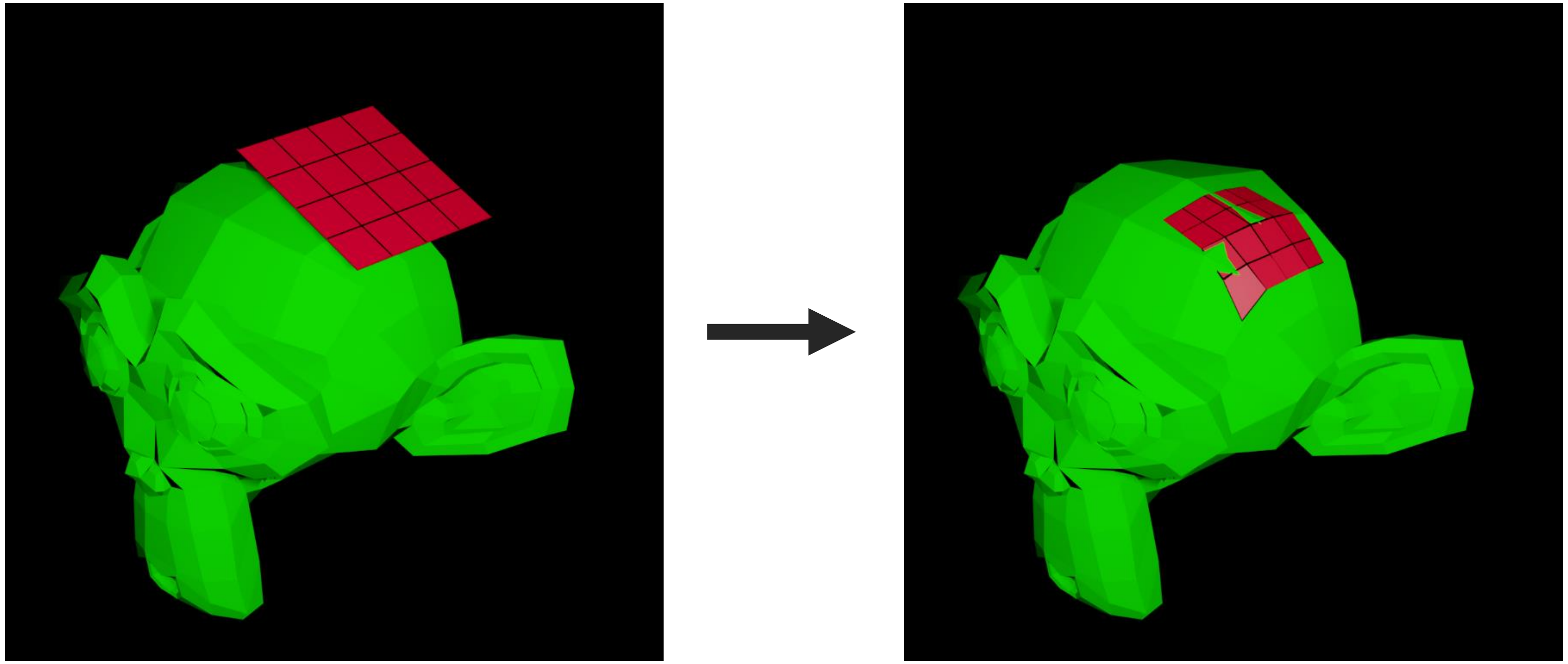
Decals: Approach 1

- What if we try it with more complex geometry?



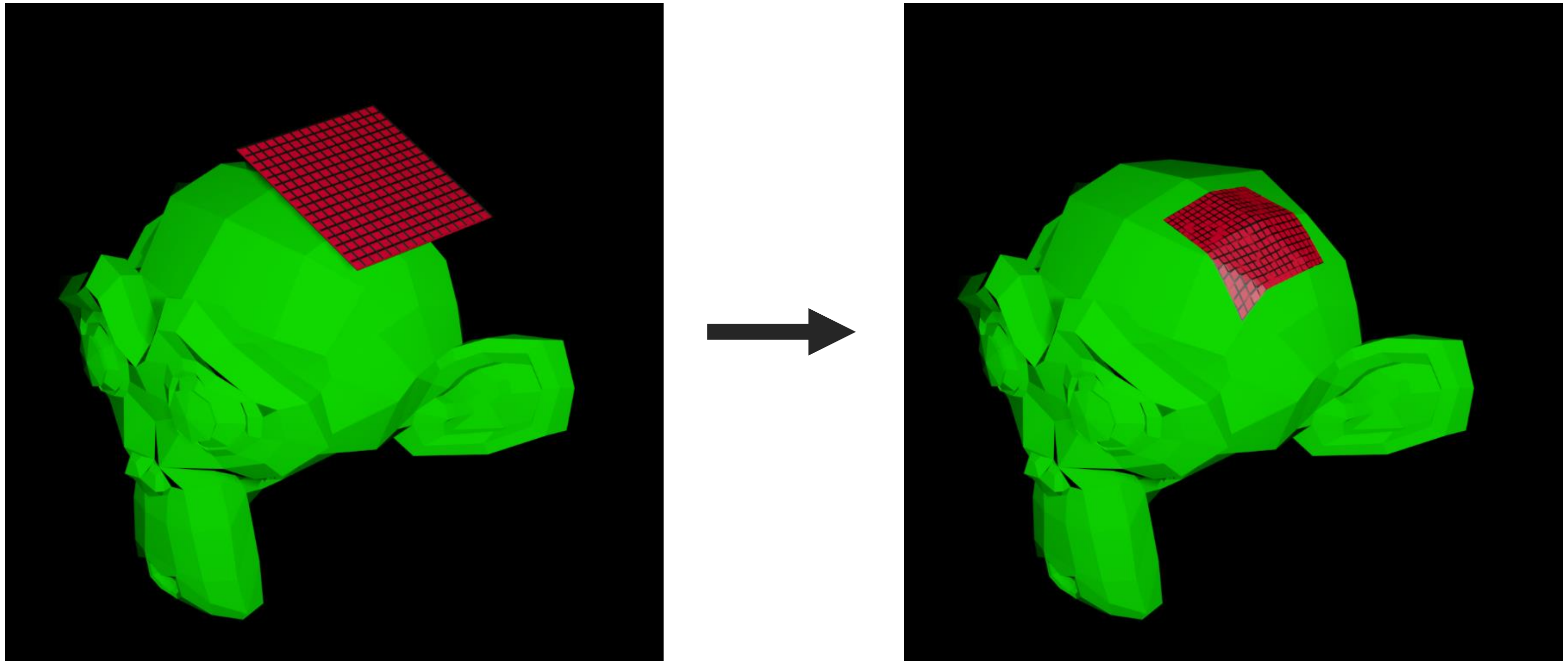
Decals: Approach 1

- Lets try subdividing the quad…



Decals: Approach 1

- Lets try subdividing the quad even more…



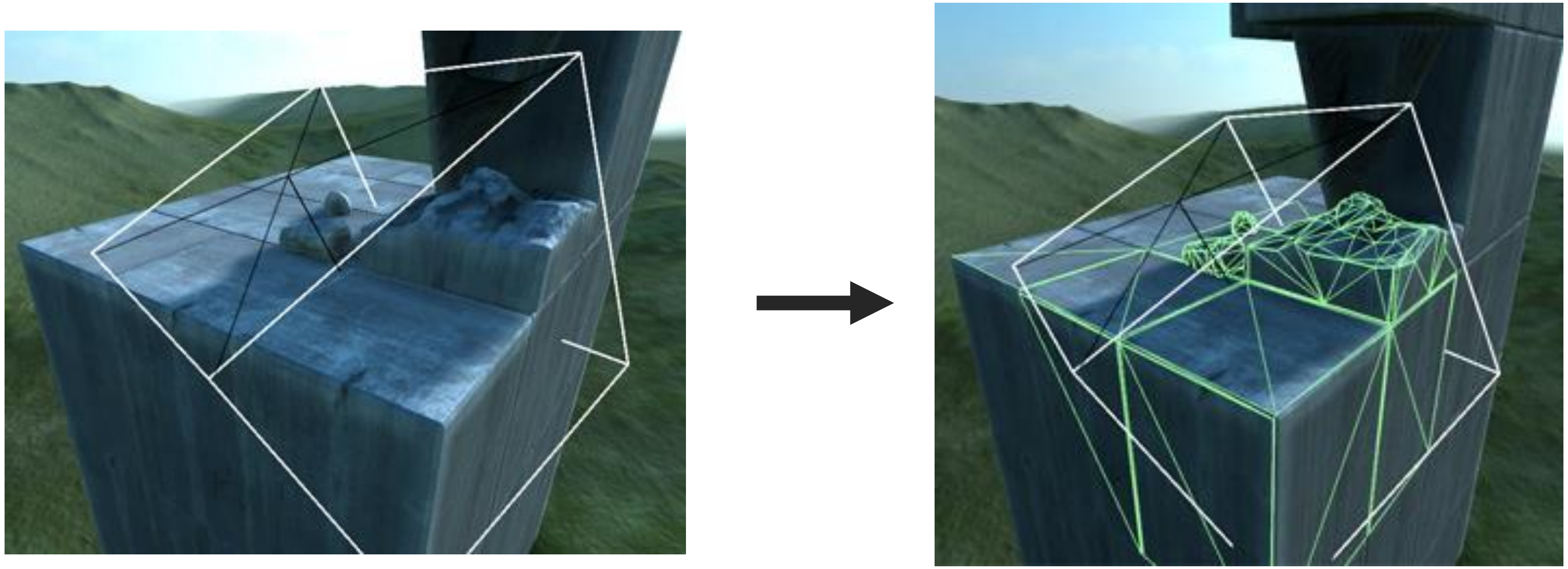
Decals: Approach 1

- This approach implies adjusting the number of subdivision depending on the projection
  - Good for static decals
  - Not so much for the dynamic ones…
- Geometry is generated at runtime, which has a performance cost
  - New geometry might need lots of vertices
- Lighting is computed twice for the occluded part

Decals: Approach 2 (Volume Decals)

- Reverse the logic of approach 1
  - No more projection of the quad onto the surface
  - Define projected volume and get every piece of mesh that lays inside
  - Intersect scene against “projector” (box shaped)

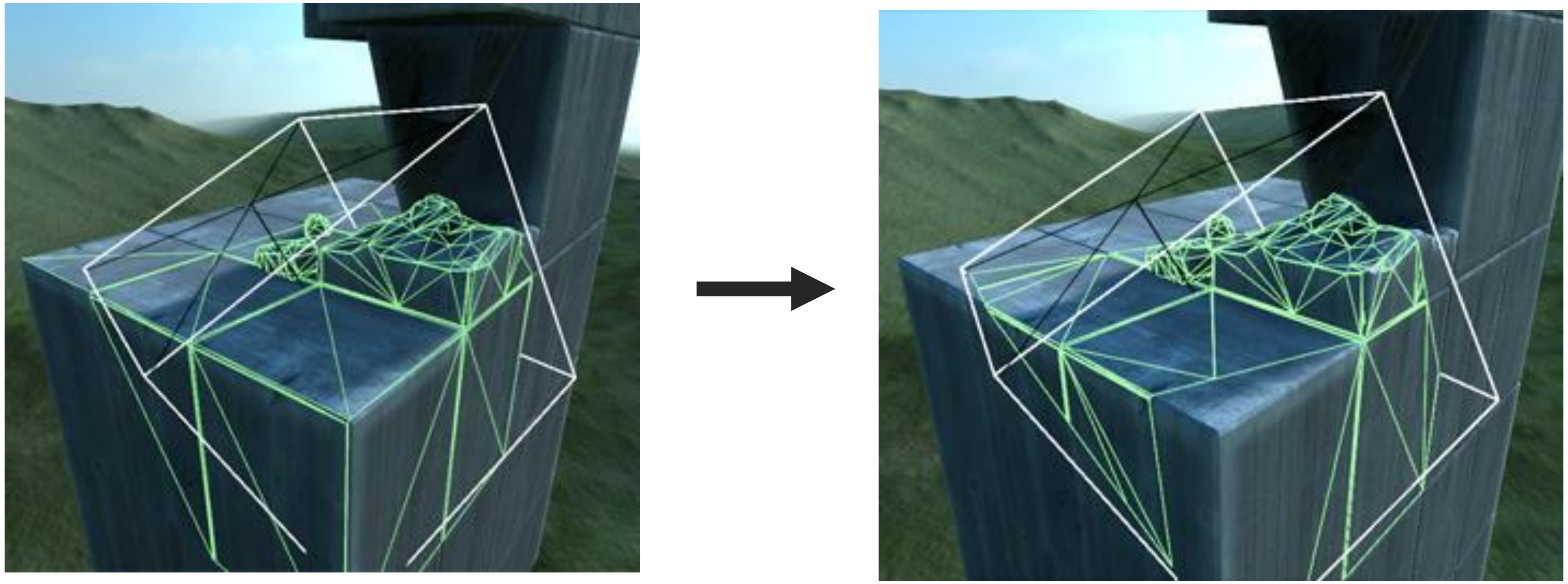
Volume Decals



Volume Decals

- Some triangles will be partially inside
  - We have to crop those triangles
  - Transform the triangles in the scene to projector space
    - Inverse of the projectors model matrix
  - Clip vertices that are outside the boundaries (0,0,0) and (1,1,1) or (-0.5,-0.5,-0.5) and (0.5,0.5,0.5)

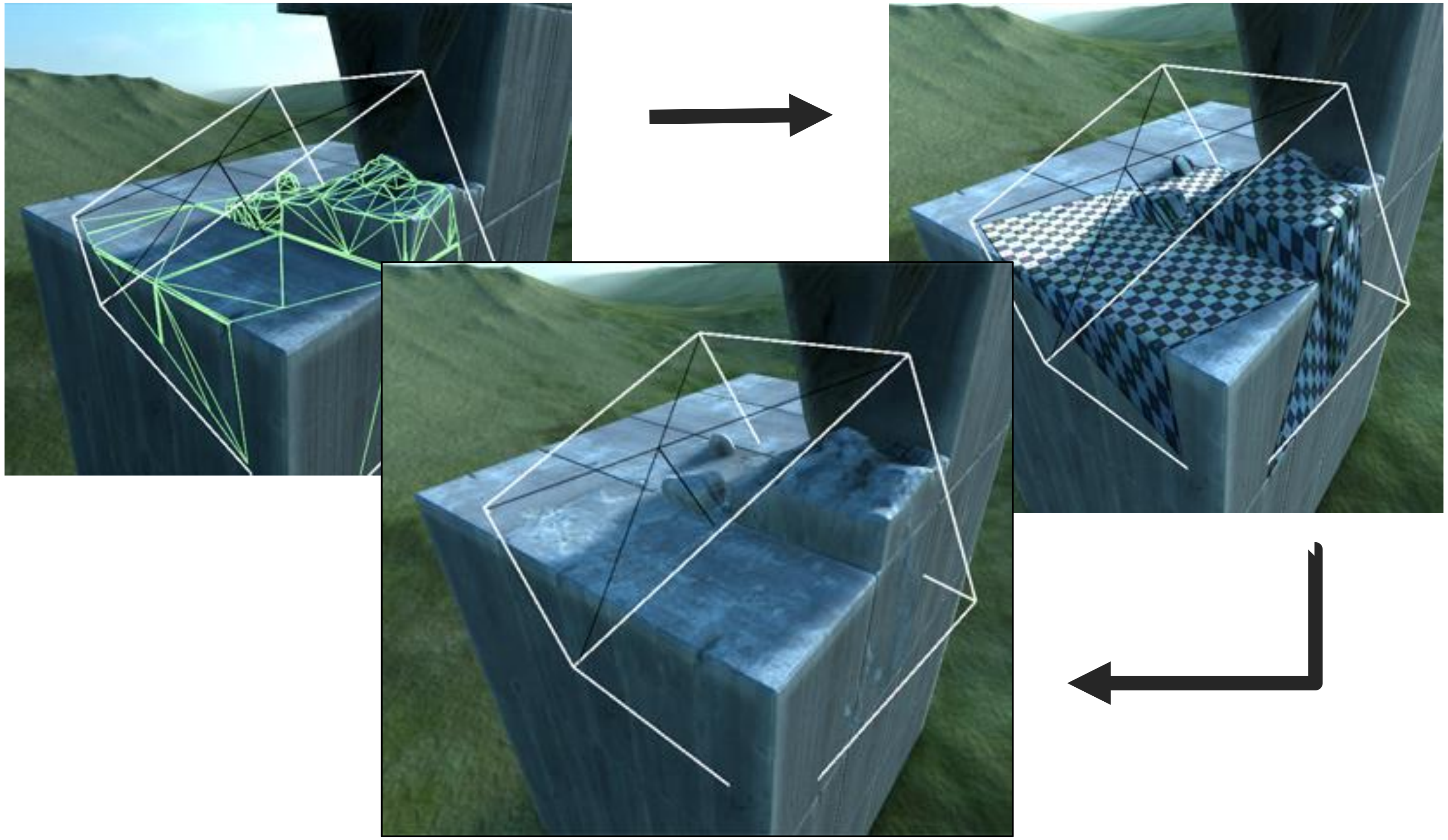
Volume Decals



Volume Decals

- When computing the intersection, the projector space coordinates also work as texture coordinates to sample the textures

Volume Decals



Volume Decals

- The algorithm depends on the scene complexity, the projected decal is irrelevant
  - Can take advantage of space partitioning
- Easy to work with for artists
  - Just placing cubes on the scene
- Geometry is generated at runtime, which has a performance cost
  - New geometry might need lots of vertices
- Lighting is computed twice for the occluded part

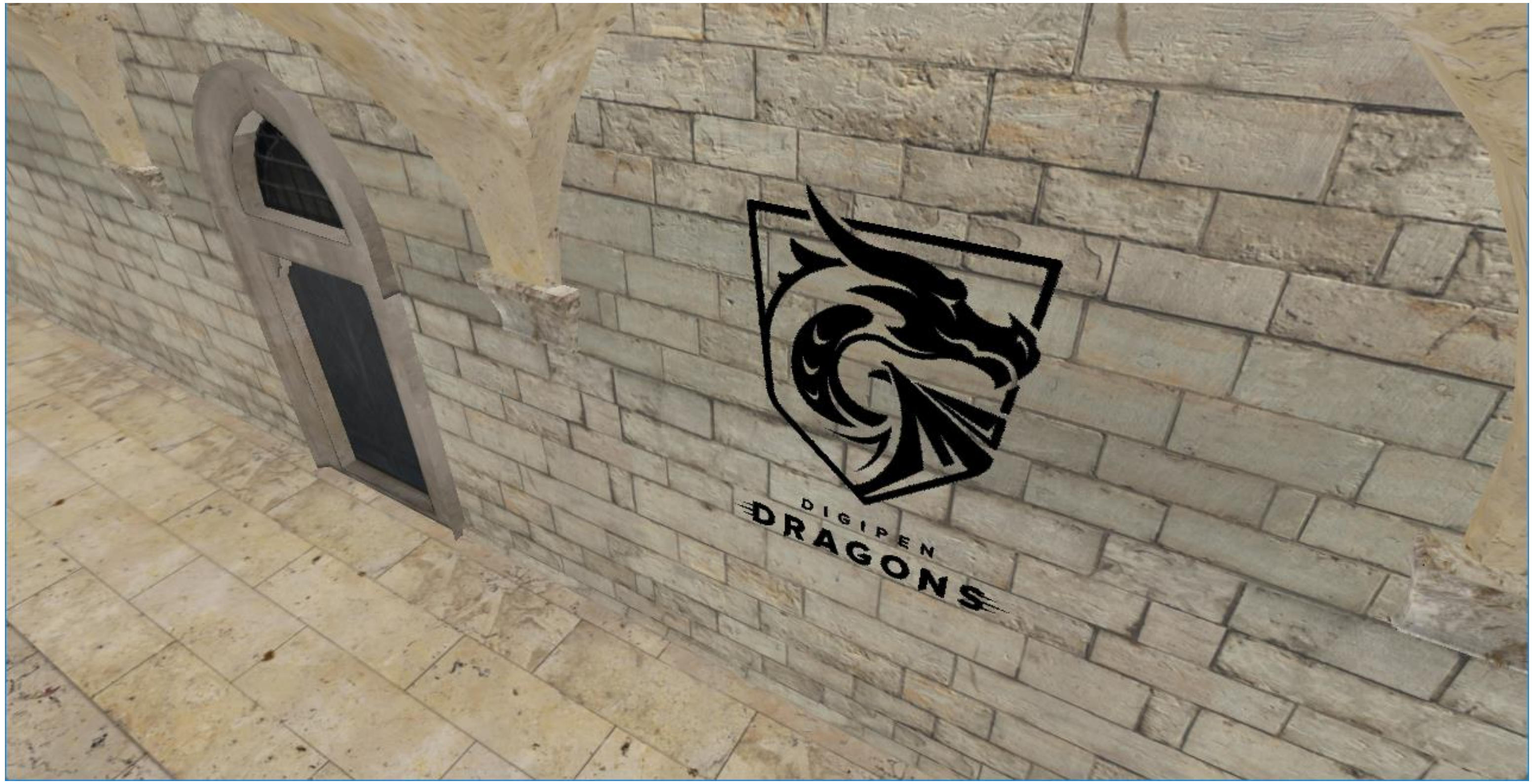
Decals: Approach 3 (Screen Space Decals)

- Relies on deferred rendering and the Gbuffer
- Adds an extra pass to the usual deferred
  - Geometry Pass
  - Decal Pass
  - Lighting Pass
- It will take advantage of the data contained on the Gbuffer and modify it for the lighting pass

Screen Space Decals

- Decal pass will consist on rendering decal volumes
  - Similar to the ones we saw in the Volume Decals approach:
    - Cubes defining the volume in which the decal will be applied
  - For each of the pixels the volume is rasterized to:
    - Read depth value from the Gbuffer
    - Calculate 3D position from that depth
    - Transform that position to the volumes’ model coordinate
    - Check if it is inside, discard otherwise
    - Use model coordinate as texture coordinate
    - Sample texture and write them to diffuse buffer

Screen Space Decals



Screen Space Decals

- Calculate 3D position from that depth
  - Convert fragment coordinate (in pixels) to NDC coordinates
  - Undo all the steps of the pipeline till world space
- Transform that position to the volumes model coordinate
  - Use the inverse of the cubes model matrix

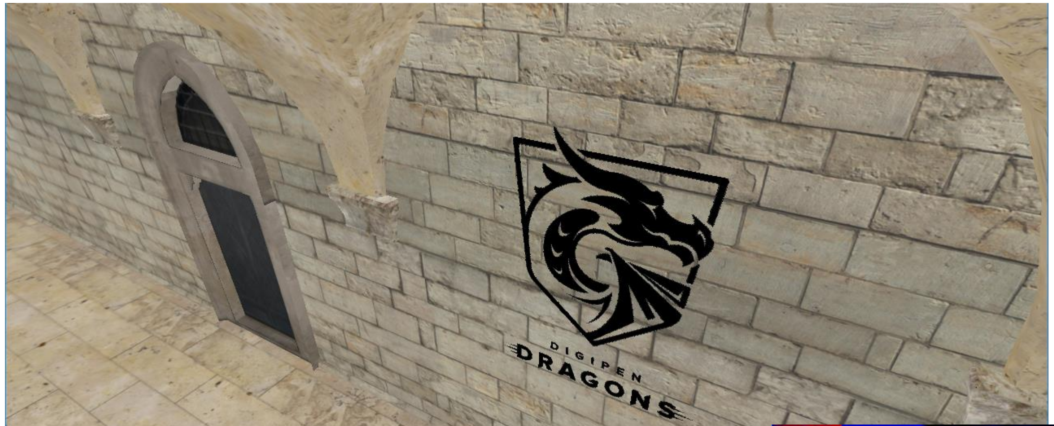
Screen Space Decals

- Lighting will be computed as usual in the lighting pass, since we are modifying the Gbuffer
- What about the normals?
  - The decal usually has its own normal map
  - Till now that was not a problem, because we were rendering a mesh with the geometry (its own tangent space)

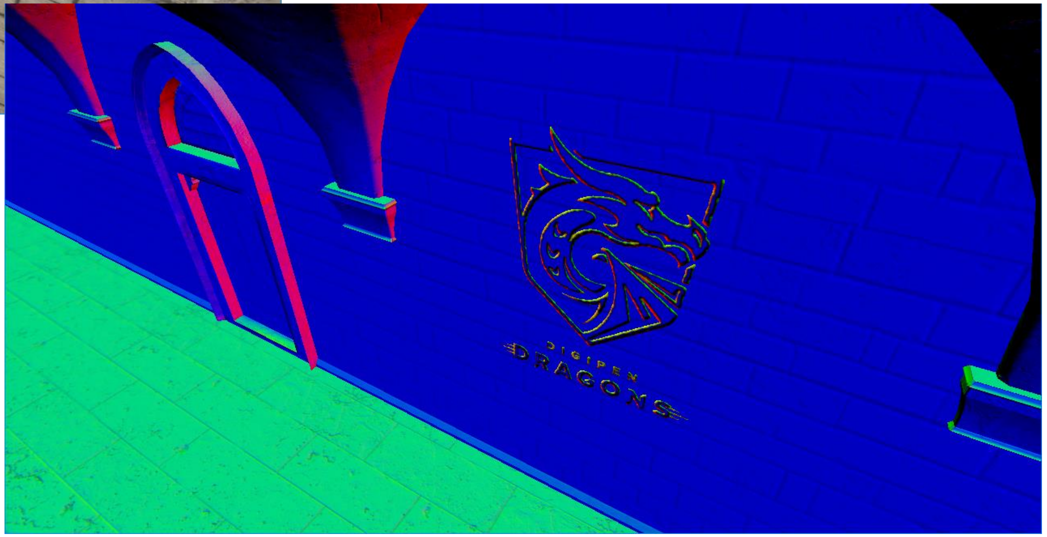
Screen Space Decals

- How can we generate a tangent space to sample the normal map of the decal?
  - Checking how the view space position is changing per pixel
  - Look up the GLSL documentation for the following functions: dFdx and dFdy
  - Those functions will help you get the tangent and bitangent (surface vectors)
  - Cross product between those two will give you the normal also in view space
  - Convert normal from map to view space
- Normal using this functions is extracted from geometry triangles and not interpolated!

Screen Space Decals



Modify the diffuse texture



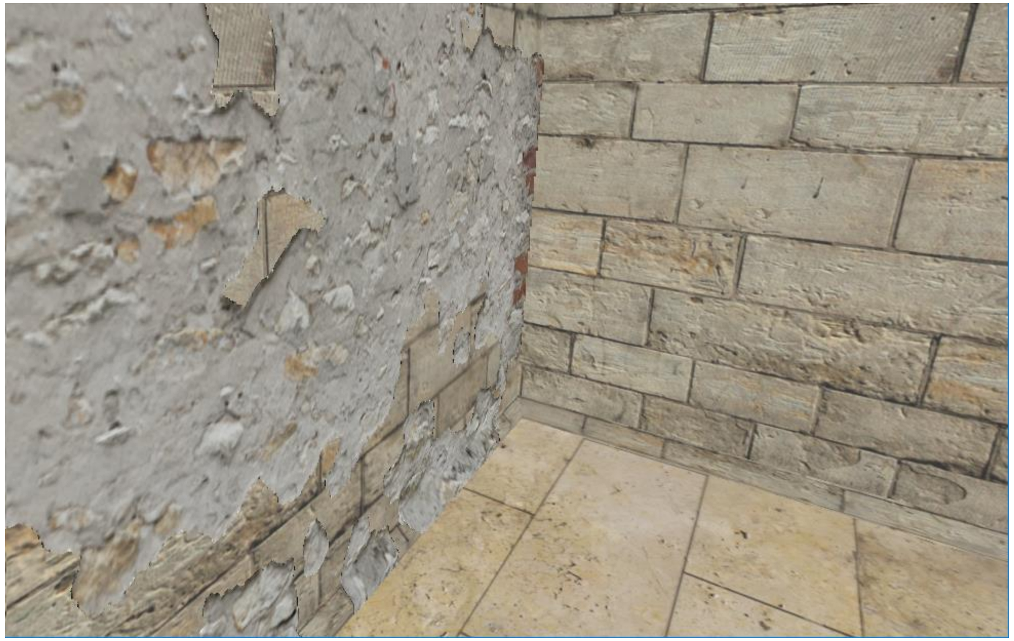
Modify the normal texture

Screen Space Decals

- Lighting is now free! But…
- Be careful with your framebuffers!
  - Cannot read/write from/to the textures attached to the framebuffer simultaneously
  - You will need a new framebuffer
- Does that mean that I will need to duplicate the Gbuffer?
  - No, textures can be attached to different framebuffers
  - Having multiple framebuffers does not necessarily imply multiple textures

Screen Space Decals

- When selecting the UV coordinates from model coordinates, artifacts will appear (side stretching)
- Solutions?
  - Discard based on the angle with the normal



Screen Space Decals

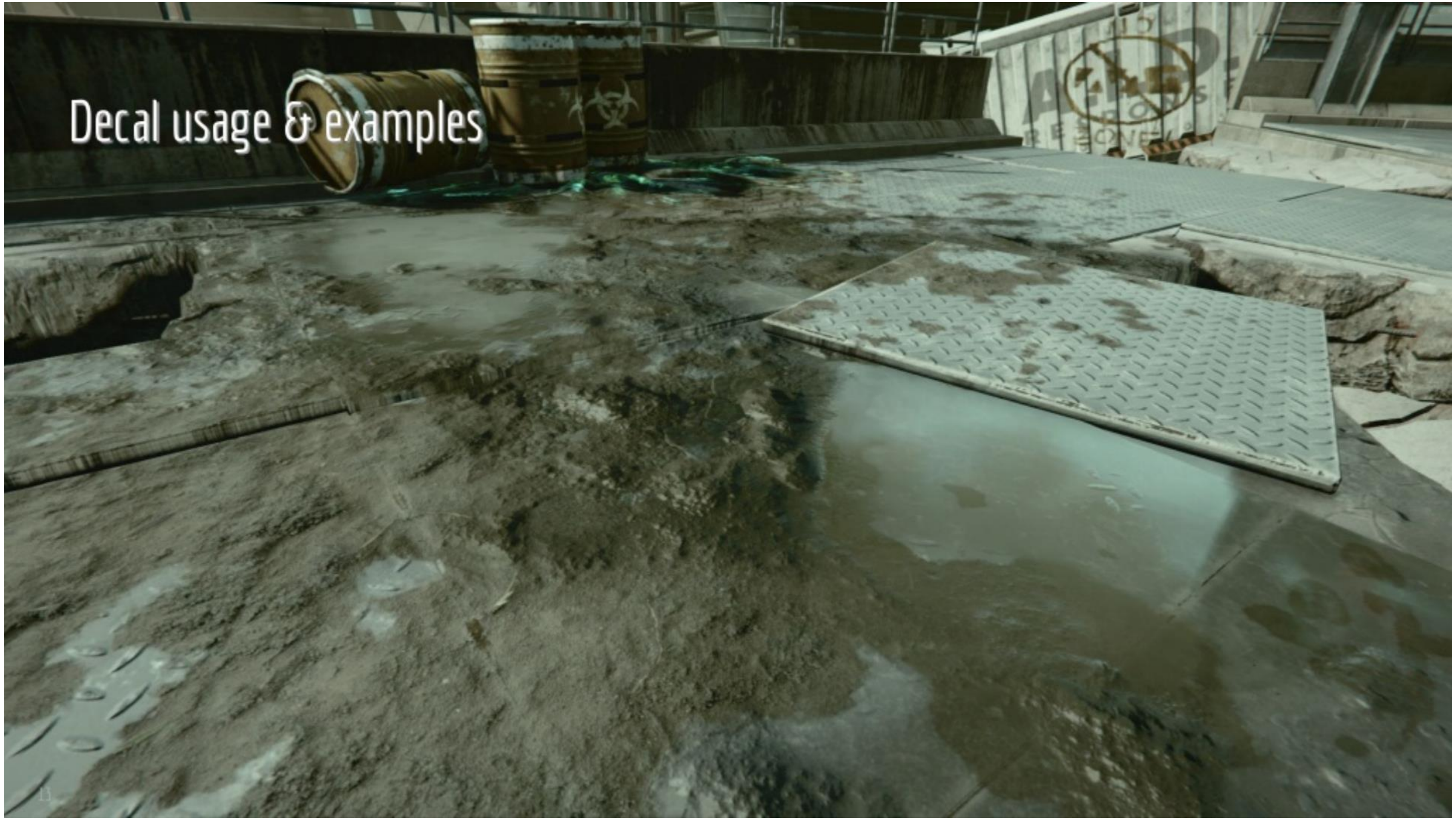
- Decal volumes will be culled and clipped when rasterizing
  - When near plane is cutting the box (clip)
  - When camera is completely inside and backface culling enabled (cull)
- Solution?
  - Render backfaces of the decal volumes

Screen Space Decals

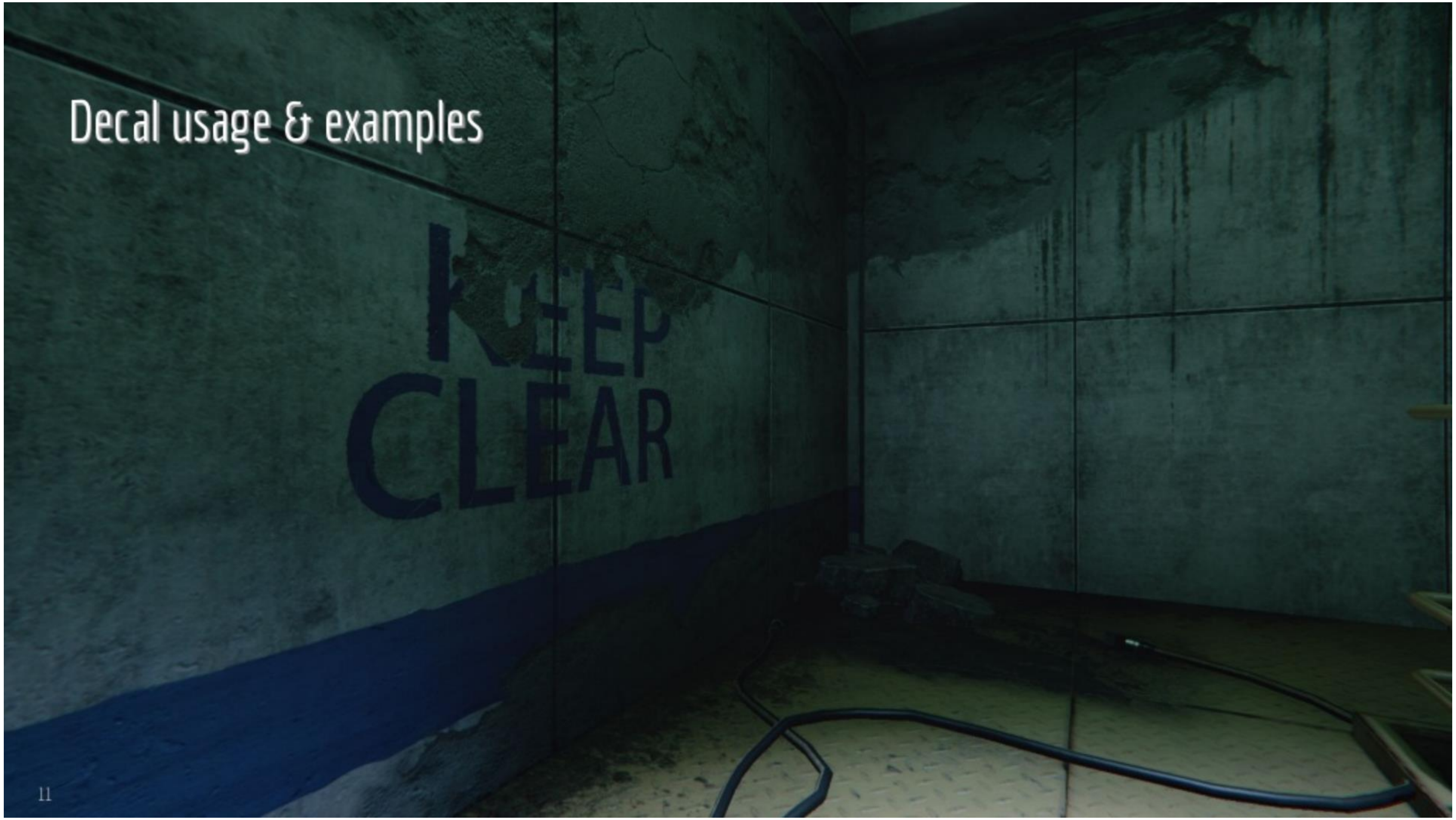
- Using huge decal volumes will affect performance
  - Rasterizer will evaluate the fragment shader more times than needed
  - Lots of wasted computation for pixels that will be discarded afterwards

Decals vs Dynamic Objects

- Mixing these two usually implies problems
- For clear hierarchical relations, decals can be attached to moving objects
  - That might not be enough with skinned meshes…
- Most of the times we want the decals to only apply to the static scene
  - Filter dynamic objects to avoid placing decals on them
  - Use stencil buffer as mask



The Surge 2



Decal usage & examples

The Surge 2

References

- http://blog.wolfire.com/2009/06/how-to-project-decals/
- Screen Space Decals in Warhammer 40,000: Space Marine, Siggraph 2012
- Bindless Deferred Decals in The Surge 2, Digital Dragons 2019
- Lighting & Simplifying Saints Row: The Third, GDC 2012
- Blender