CS562 Assignment 3: Screen Space Decals

OBJECTIVE

In this assignment you are going to implement Screen Space Decals within your deferred shading pipeline. This assignment requires you to implement normal mapping on the decals added to the scene.

1. Scene setup

Load the provided scene file that will import *sponza* model as the only object. On top of that a new element of data has been included on the scene file, the decals. Those are represented as a new list of elements that have the following format:

```
"decals":
[
    {
        "diffuse": "./data/textures/decal_crack_d.png",
        "normal": "./data/textures/decal_crack_n.png",
        "translation":
        {
            "x": 30.0,
            "y": 0.0,
            "z": 0.0
        },
        "rotate":
        {
            "x": 90.0,
            "y": 0.0,
            "z": -90.0
        },
        "scale":
        {
            "x": 60,
            "y": 40,
            "z": 1
        }
    }
]
```

Most of the data shown is similar to the regular objects, but the mesh has been substituted by two textures. That is because the mesh used by decals will always be the *Cube.gltf* and the only difference between different decals is the diffuse texture and the normal map.

2. Algorithm Overview

Decals are a great way to easily enhance and scene without adding extra geometry and break the tiling and instancing patterns. In Screen Space Decals this extra step happens between the Geometry and Lighting passes with and added Decal pass.

1. **Application:** Draw the decals loaded from the scene file after the geometry pass. You will need to create a new frame buffer for that pass that modifies the diffuse and normal. Careful with overwriting the alpha channel of those textures, because they may contain relevant data.

   The lighting pass should not be modified, but you need to make sure that the input to this stage is the update diffuse and normal buffer.

2. **Vertex Shader**
    - Usual matrix multiplication to set `gl_Position` in clip space.

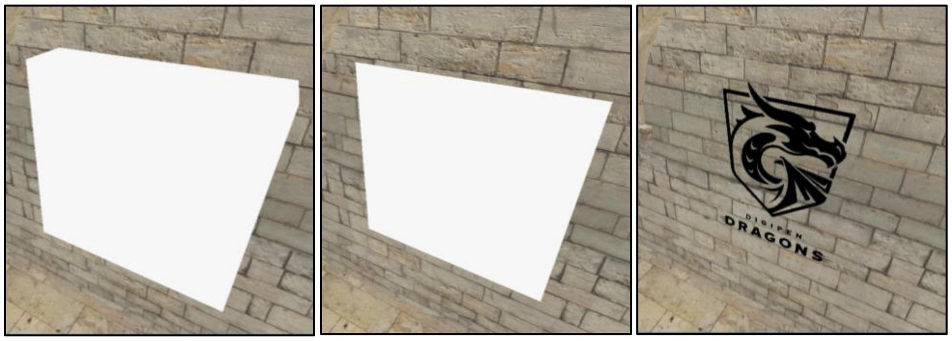
3. **Fragment Shader**

    - 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 the box, discard otherwise.
    - Check if normal is valid respect to the provided angle limit.
    - Use model coordinate as texture coordinate.
    - Sample texture and write them to diffuse and normal buffer (make sure your normal buffer data is written in the same space as all the other normals).

INPUT

With the goal of showcasing your work the best possible way the following specifications need to be meet:

- A first-person camera, controllable by the user via the keyboard and/or mouse.
    - WS: move the camera forward/backward along its forward vector.
    - AD: move the camera left/right along its side vector.
    - QE: move the camera up/down along the **global** up vector (0,1,0).
    - Click + Drag: the mouse should tilt the camera around its yaw and pitch.
    - Up/Down: rotate the camera around the pitch.
    - Left/right: rotate the camera around the yaw.
- Pressing the **Control + R** key should reload the scene from the file.
  Pressing **F5** should reload all the shaders.
- Allow the enabling/disabling of decal rendering
- Adjust angle limit respect to the geometry respect to the decal.
- Three render modes for the decals:
    - Render full decal volume as white (use any normal for the normal map).
    - Render only pixels that are part of the geometry that fall inside the decal volume (use any normal for the normal map).
    - Render decal textures with the proper shading.



COMMAND LINE ARGUMENTS

The program should allow for the following arguments passed to the main function through the command line arguments.

- `argv[1]`: JSON scene name to load
  - Can be omitted, a default scene (the one provided) should be loaded in that case
- `argv[2]`: Filename of the screenshot to save at the end of the first frame in PNG format
  - Can be omitted and no screenshot is saved in that case
  - Can only be used if the scene is also specified

Example: `cs562_jon.sanchez.exe scene.json screenshot.png`

SUBMISSION

The project will be submitted as a Visual Studio 2019 that will compile and execute without any adjustment. The solution will **only** have the **64-bit platform**, so the 32-bit configuration should be removed. The framework will be created from scratch and will have the following folder hierarchy:

- data -> Folder with all the resources
  - gltf -> Meshes and textures to load **(NOT included on submission)**
  - scenes -> Scene json files **(NOT included on submission)**
  - shaders -> Implementation of the shaders **(INCLUDED in submission)**
- src -> Folder with your code
  - Any header only library to compile (ImGui, stb…)
- cs562_<login>.vcxproj
- cs562_<login>.vcxproj.filters
- cs562_<login>.vcxproj.user
- Required DLLs (only x64 versions)
- include (all include folders for the libraries used)
- lib -> All library files for the libraries used)
- obj -> Folder where the compilation intermediate files when solution is compiled **(NOT included on submission)**
  - Debug
  - Release
- bin -> Folder where the executable will be generated when solution is compiled **(NOT included on submission)**
  - Debug
  - Release
- Solution file (cs562_<login>.sln)
- README.txt -> Text file containing header (see below) as well as controls for the application.

README FILE

Along with your Visual Studio 2019 solution you will attach a README.txt file that will contain the following contents:

1. **How to use your program:** Every control available when executing the program and hints on how to get use it.
2. **Important parts of the code:** Specify source code files and approximate function or lines that were most relevant for the assignment.
3. **Known issues and problems:** Any not completed part of the assignment or known bugs that can be encountered when running the program.

GRADING SCHEME

Section	Item	Points
Decal	Decal pass	10
	Position reconstruction	15
	Normal reconstruction	15
	Correct framebuffer	5
	Volume intersection	10
	Correct UV extraction	5
	Angle filter	5
Controls	Flexible camera	5
	Enabling/disabling decals	5
	Adjusting limiting angle	5
	Decal volume rendering	5
	Projection inside decal rendering	5
	Shaded drawing mode	10