Parallax Mapping

Travis Chan, Evan Molinares, James DeLoach

CS 4204 Computer Graphics

Final Project

Overview

Parallax mapping is a rendering technique that enables the appearance of depth where there is none.

Rather than adding thousands of vertices, this method uses a heightmap to indicate the depth of regions on a plane, which are then evaluated in the fragment shader.

Typically, this reduces strain on the CPU and increases performance compared to complex meshes.

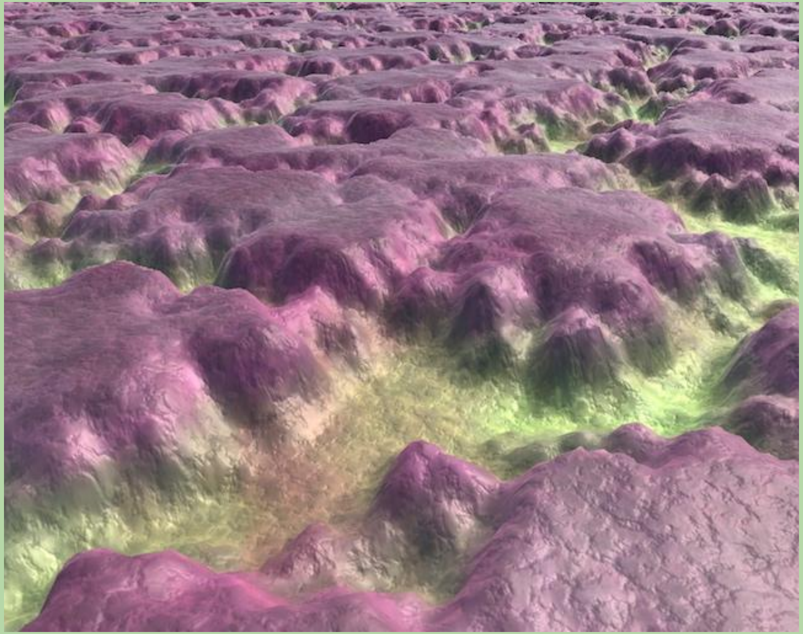


Figure 1. Parallax mapping with normal mapping and extra techniques. The rendered mesh in the image has a total of two triangles. [1]

Motivation

Our group implemented a version of Parallax Mapping, where a specific “height” texture is used to define the level of displacement of a particular texel to achieve a higher visual quality than normal mapping alone.

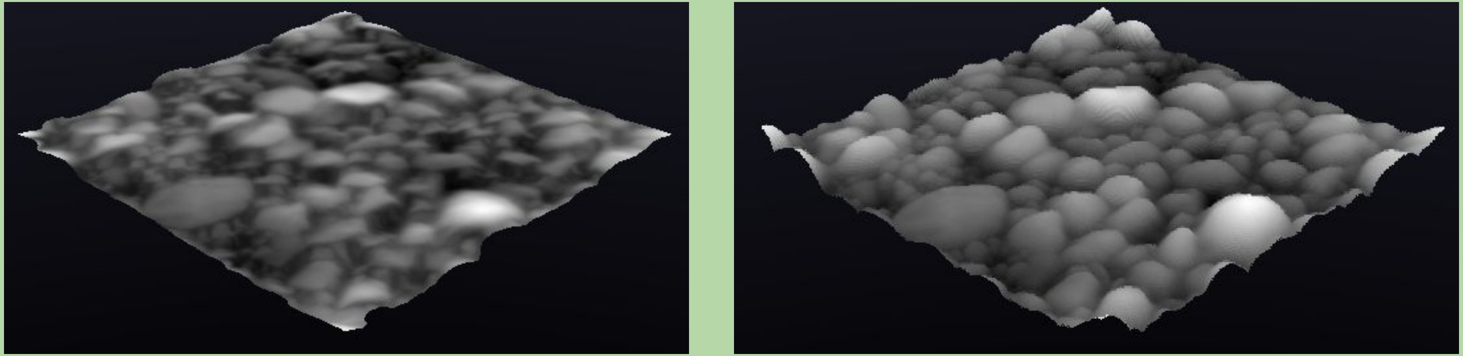
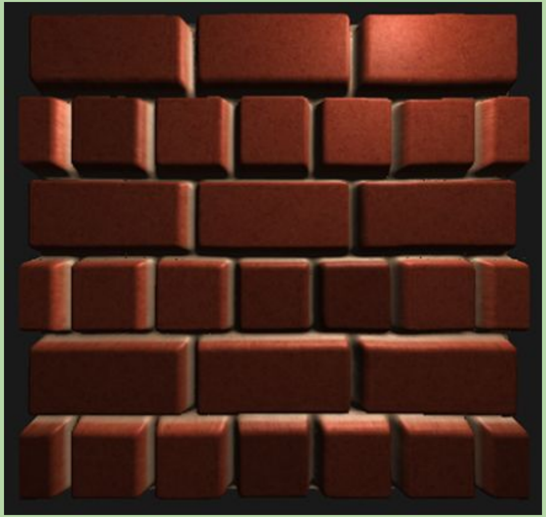


Figure 2. Left: Normal mapping of a plane, next to no depth effects visible. Right: Parallax mapping of the same plane with clear peaks and valleys. [2]

Methodology

In order to implement Parallax Mapping, we had to implement texture mapping into our graphics pipeline. To facilitate this, we had to revise our model importer to accept gltf/glb files, which define vertex normals as well as UV maps which made reading UV coordinates significantly easier.



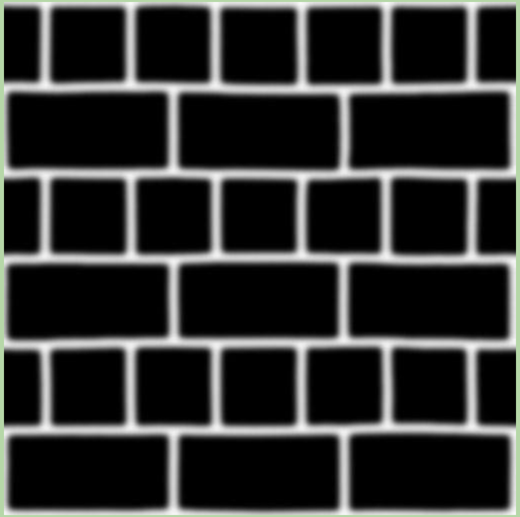
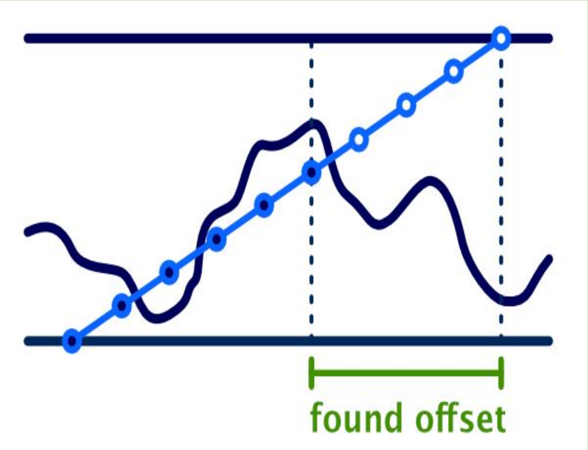


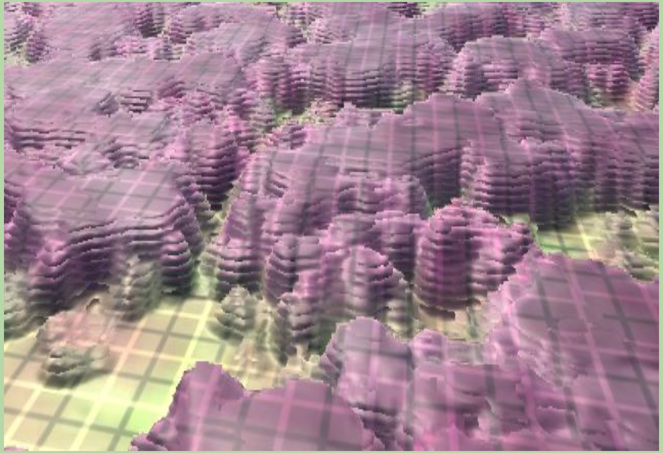
Figure 3.
Left: Parallax mapping of a brick wall. The mesh is a single plane.
Right: Height map of the wall, ranging from 0.0 (black) to 1.0 (white). [3]

Methodology

Most implementations of Parallax Mapping take place completely within UV space (with view direction information). We took a different approach by adjusting the depth buffer directly as our pipeline already allows for fairly easy manipulation of this buffer that other pipelines generally can’t affect directly. This also circumvents a traditional issue with parallax mapping shown below.

Figure 4. Left: Raymarching implementation of steep parallax mapping. Each fragment samples the texture 10 times, and detects changes in the heightmap, a costly operation. [1] Right: The resulting image.





Methodology

To demonstrate successful integration of Parallax Mapping, we rendered a brick wall [3] and components from Mario Super Bros, specifically a brick block, mystery box, coins, and Mario himself [4]. Each object contains a unique depth feature that will showcase parallax mapping. Height maps were generated for the objects [5].

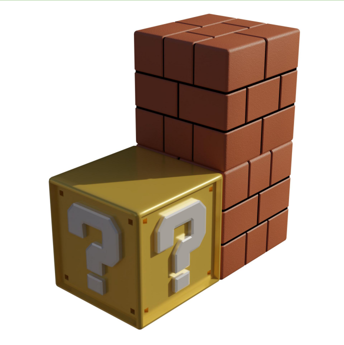
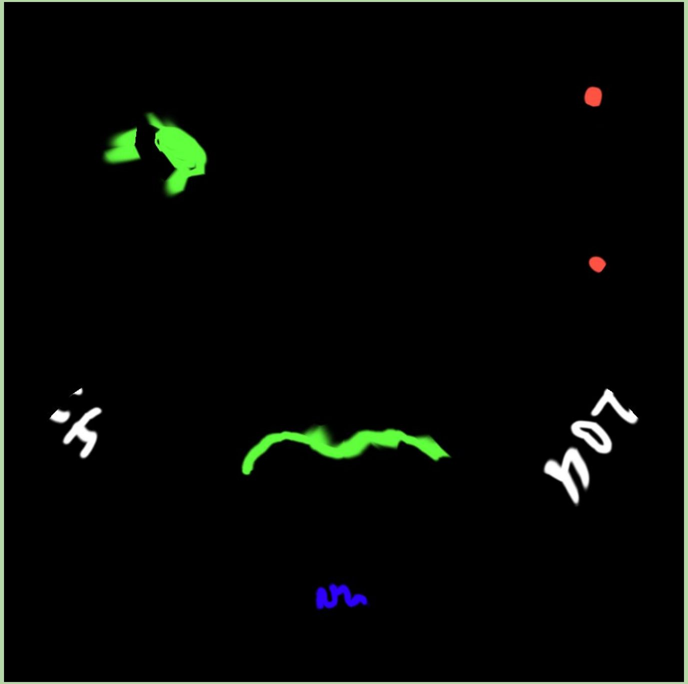


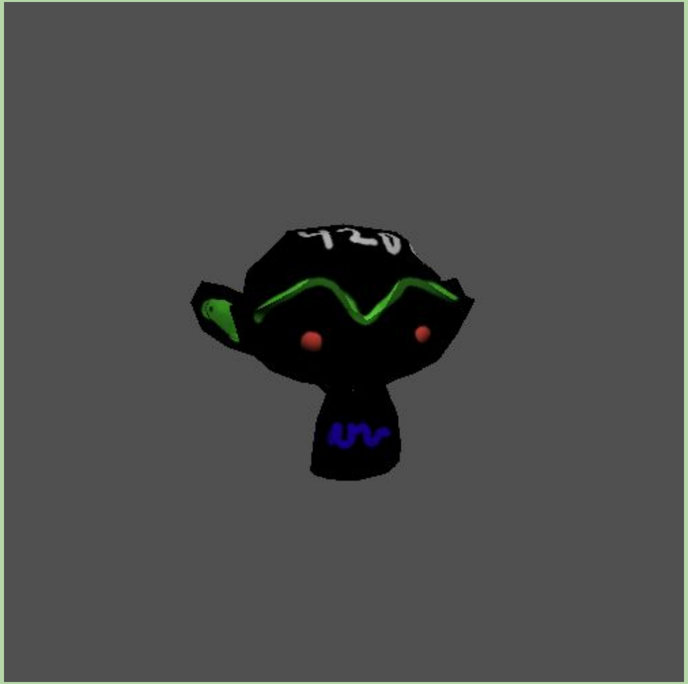
Figure 5. Brick block and mystery box from Super Mario Bros that will be rendered using parallax mapping [6]

Implementation Details

Reading in Model Data From GLB Files

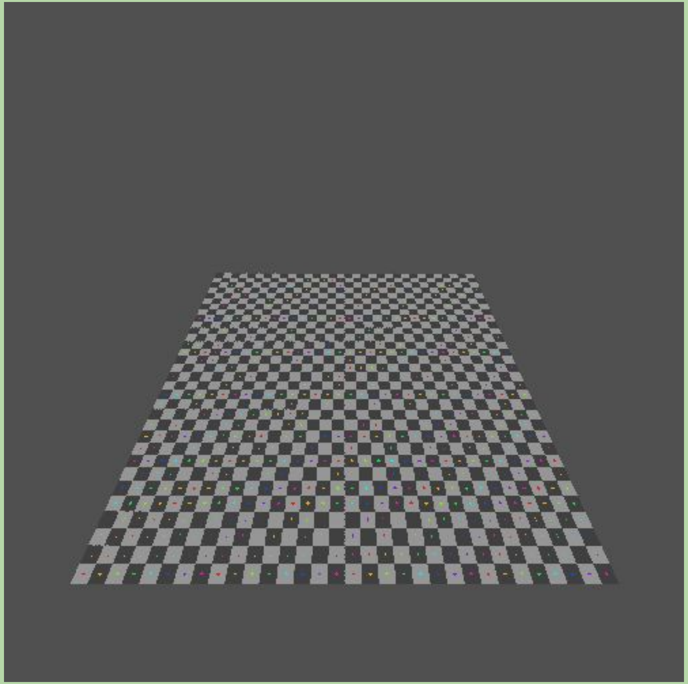


Test texture map



Import test output

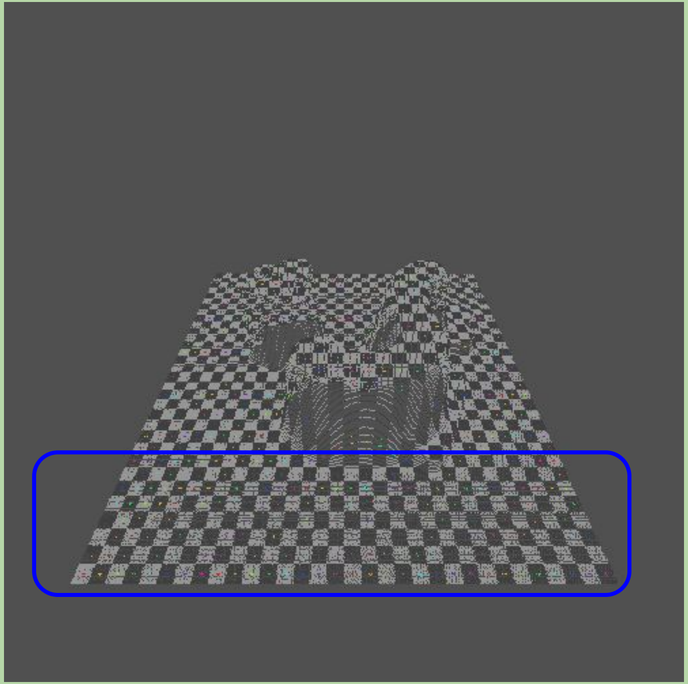
Perspective Correct & Height Map



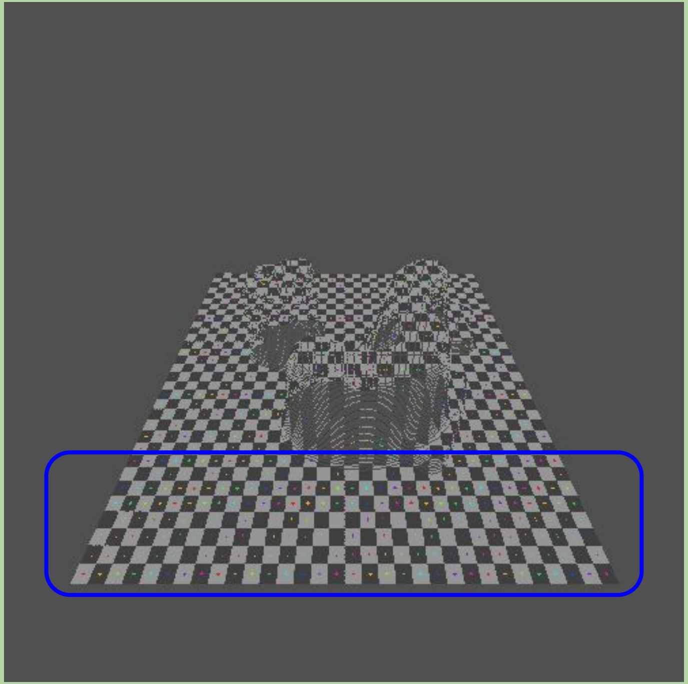
Perspective correct uvs

Height map

Parallax Mapping Artifacts

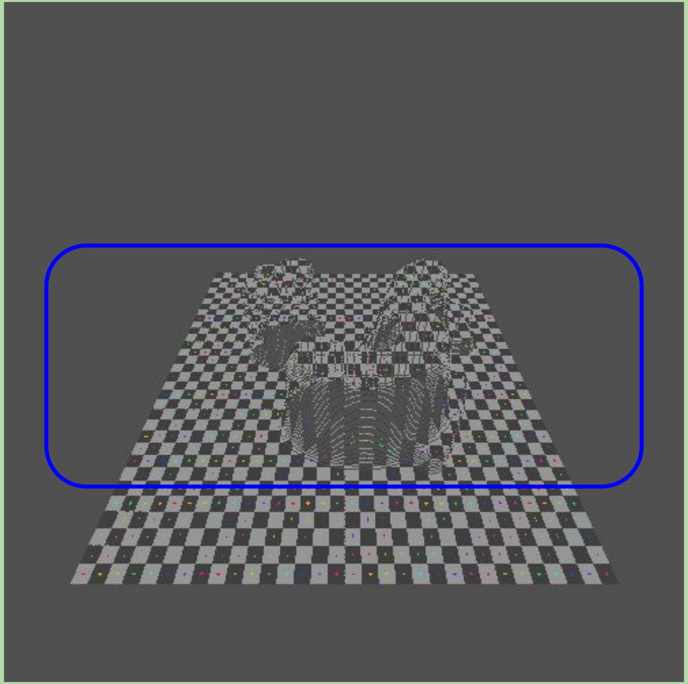


Initial Artifacts

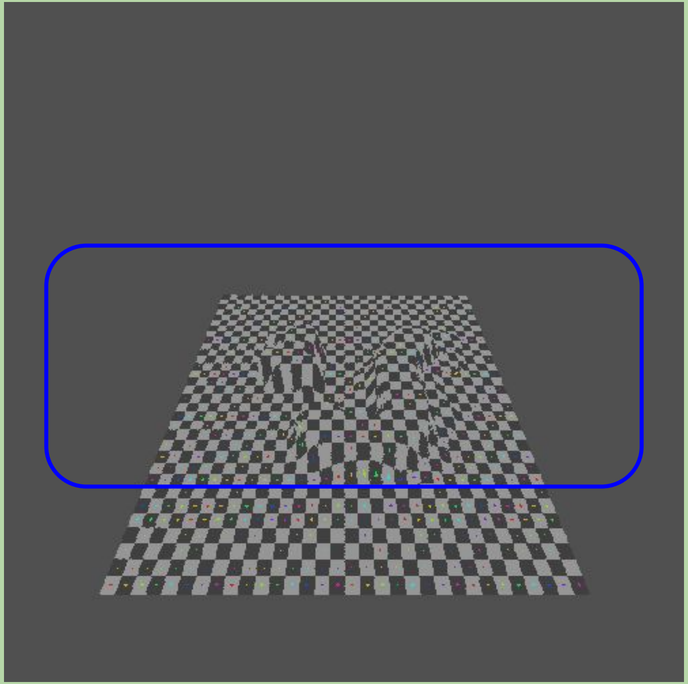


Pixel Center Offset

Parallax Mapping Artifacts



Layer Banding

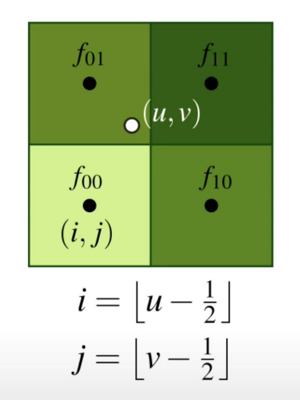


Depth Buffer Interpolation

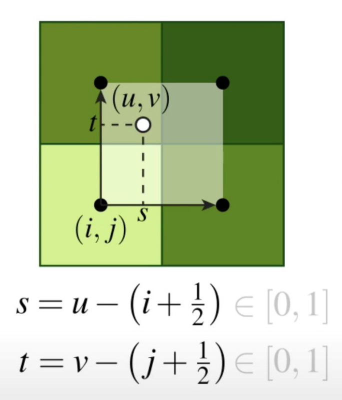
Bilinear Filtering

Bilinear filtering is an interpolation technique utilized to blend colors [7]

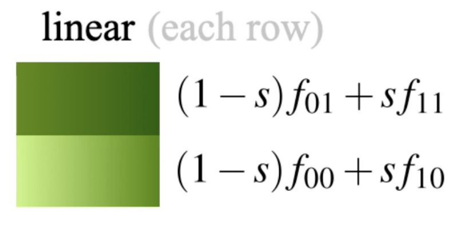
1. Identify four closes texels (texture pixels)
2. Interpolate uv coordinates relative to the center of each of the four texels
3. Blend in the horizontal direction
4. Blend in the vertical direction



$$i = \lfloor u - \tfrac{1}{2} \rfloor$$
$$j = \lfloor v - \tfrac{1}{2} \rfloor$$

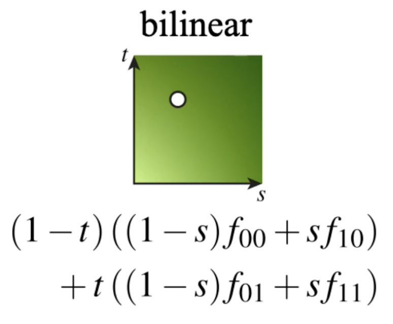


$$s = u - (i + \tfrac{1}{2}) \in [0,1]$$
$$t = v - (j + \tfrac{1}{2}) \in [0,1]$$



linear (each row)

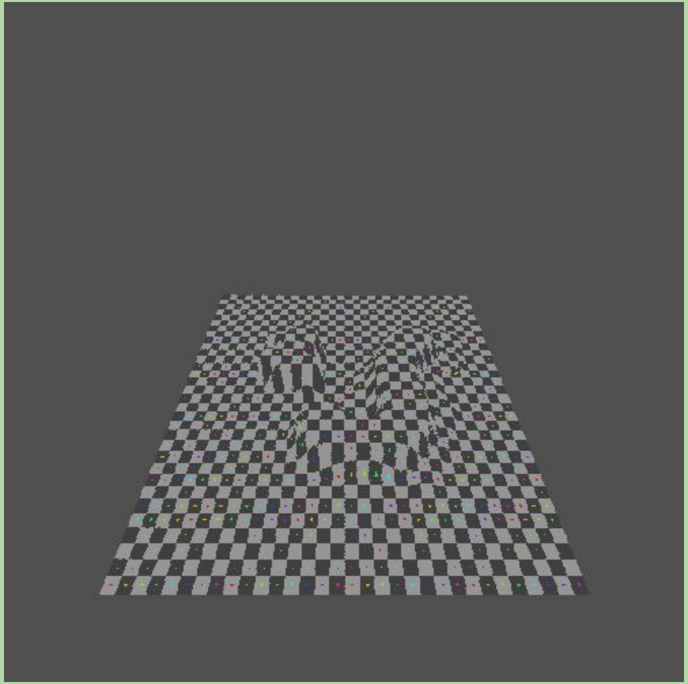
$$(1-s)f_{01} + sf_{11}$$
$$(1-s)f_{00} + sf_{10}$$



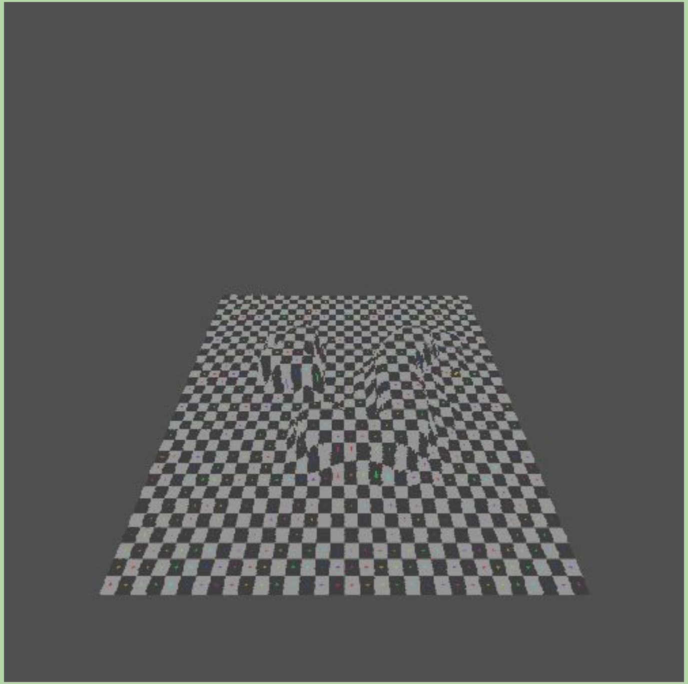
bilinear

$$(1-t)((1-s)f_{00} + sf_{10})$$
$$+t((1-s)f_{01} + sf_{11})$$

Bilinear Filtering

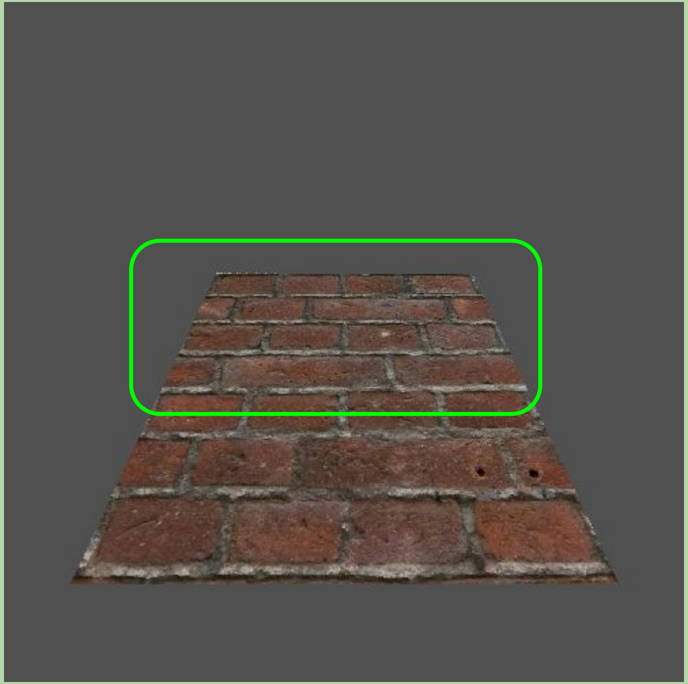


Before bilinear filtering

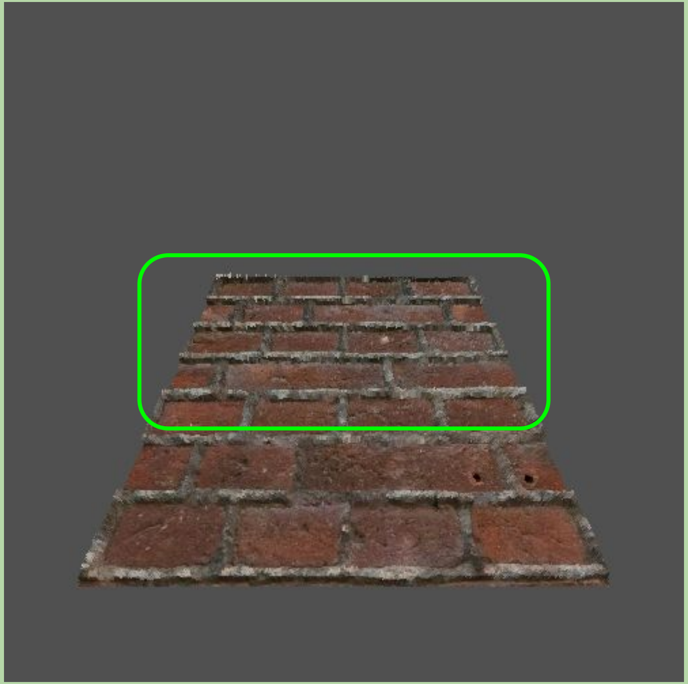


After bilinear filtering

Brick Wall Test

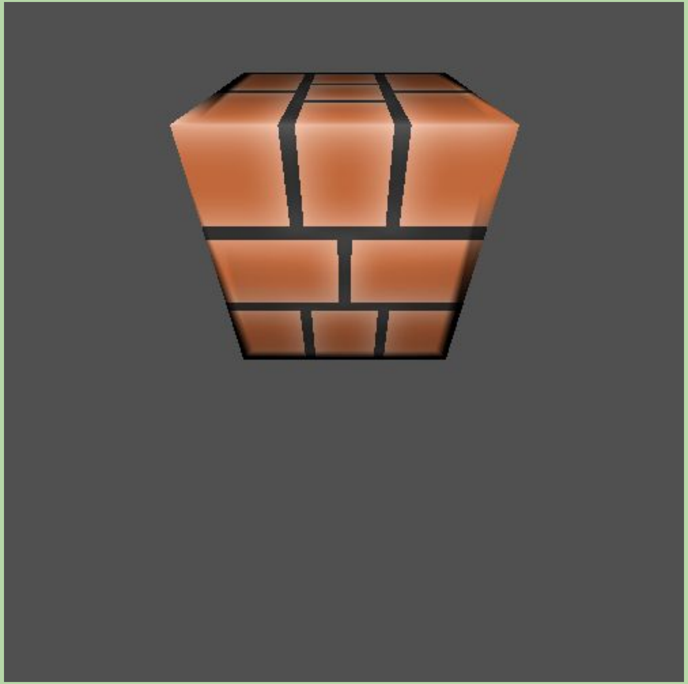


Normal mapping

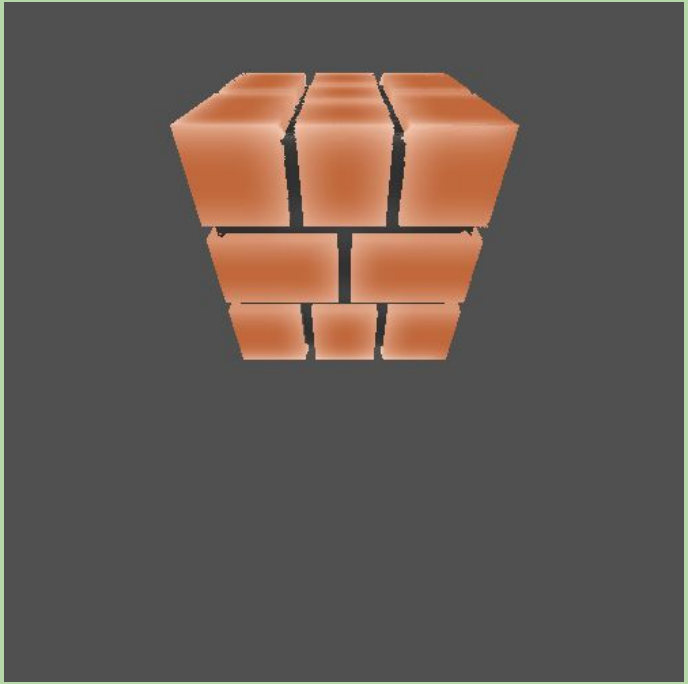


Parallax mapping

Brick Cube Test



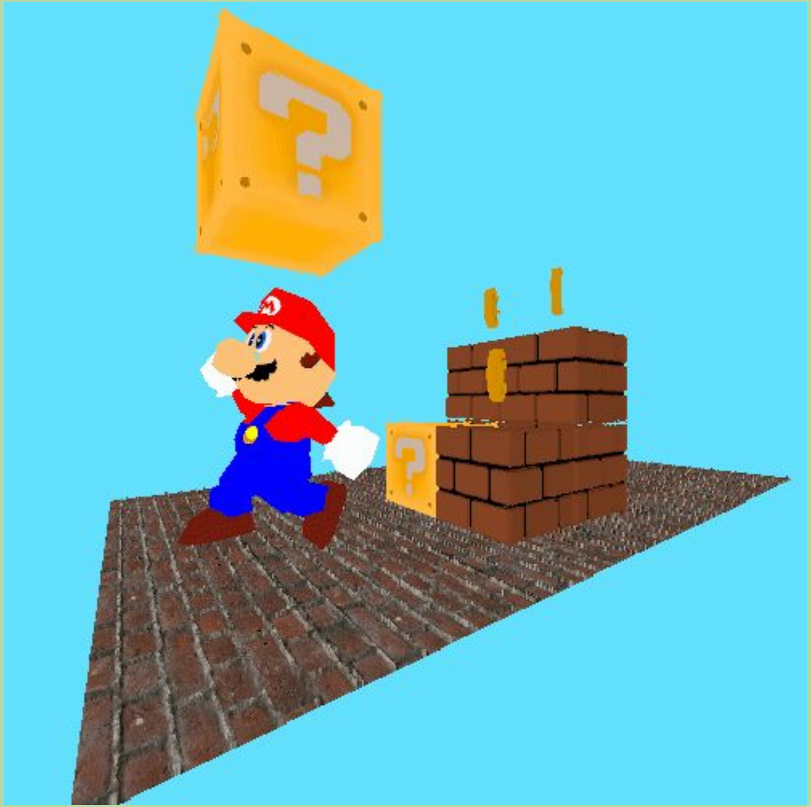
Diffuse mapping only



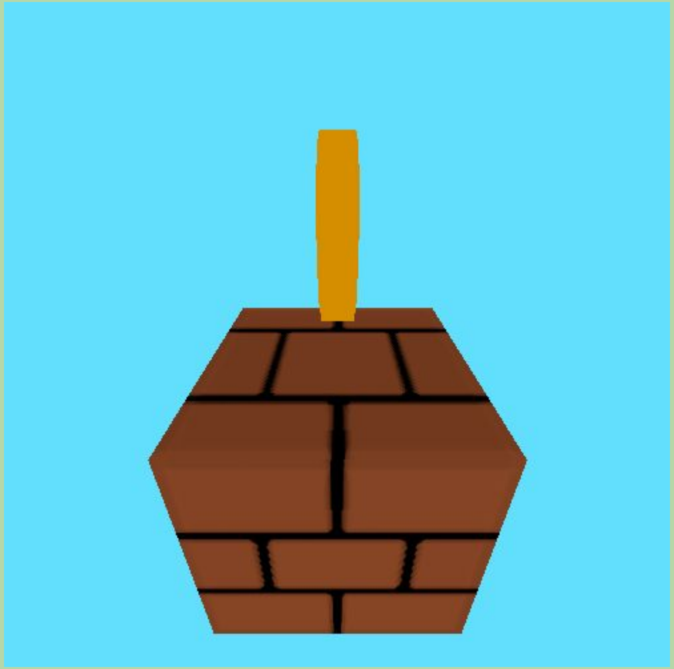
Parallax mapping

Final Results & Issues

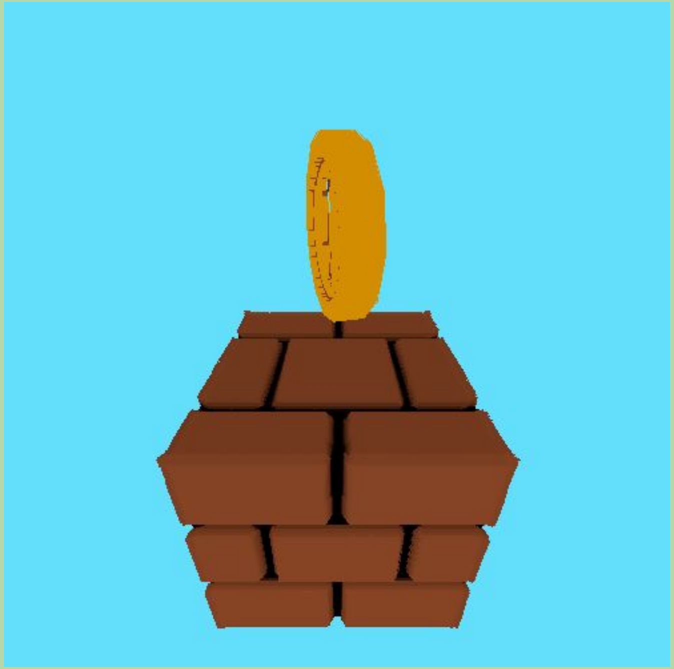
Full “Animated” Mario Scene



Spinning Coins

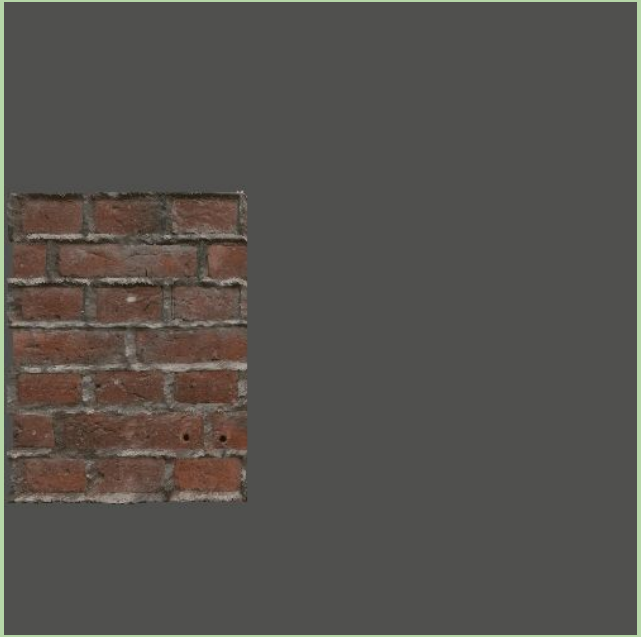


Diffuse



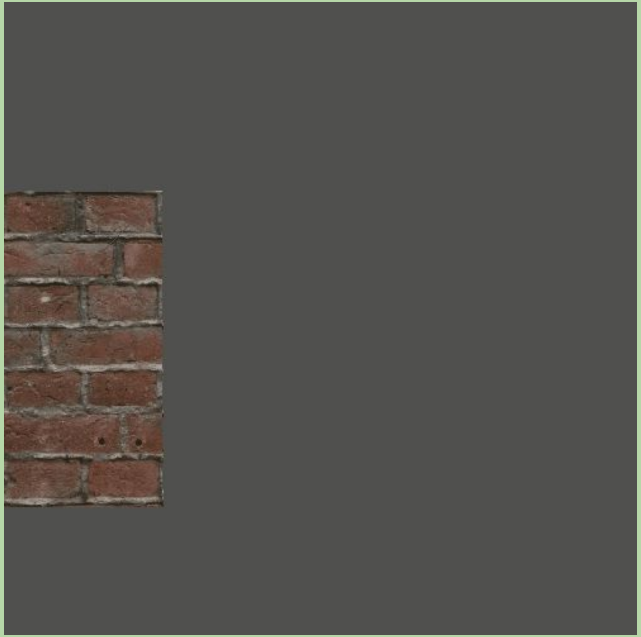
Parallax mapping

Temporal Issues

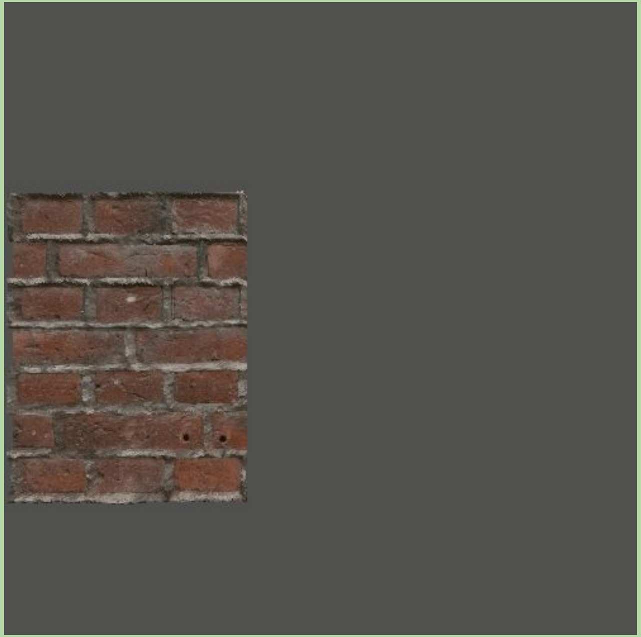


Moving Planes

Temporal Parallax vs. Diffuse

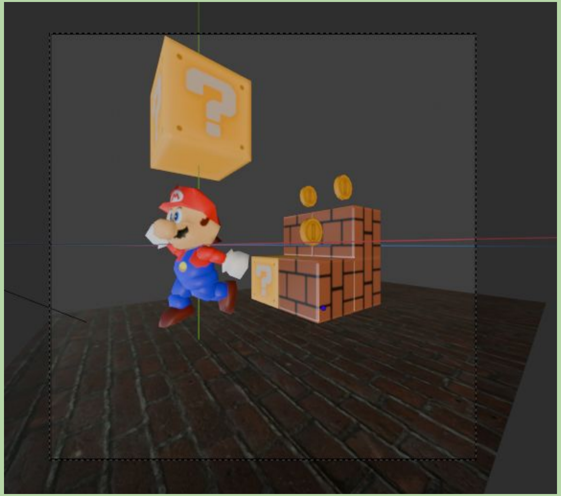


Diffuse

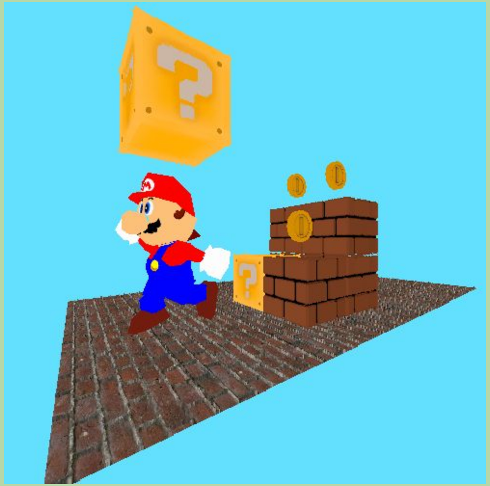


Parallax

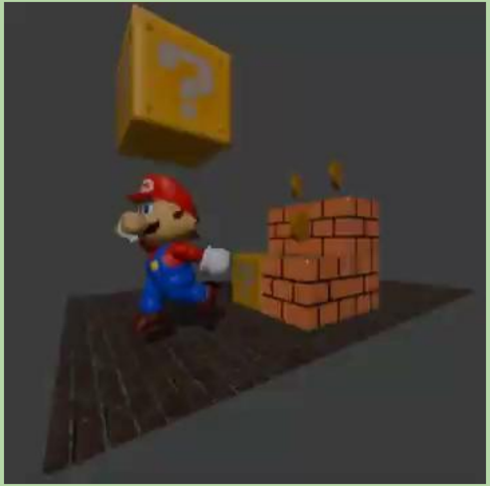
Bonus: Scene Setup



Blender Mockup



Our Pipeline



Animation Planning

References

[1] Catlike Coding Rendering 20 Parallax Mapping

[2] ShaderToy Parallax

[3] Geometrical Explanation of Parallax Mapping

[4] Super Mario Bros Wikipedia

[5] Video on How to Create Mario Block from Scratch

[6] Blocks image reference

[7] Lecture 07: Perspective Projection and Texture Mapping (CMU 15-462/662) by Keenan Crane