3D Image Rendering

Led by: Wendian Jiang (wj225)
Lin Wang (lw569)
Hanchen Jin (hj424)
Advisor: Zhiru Zhang
1. Introduction
2. Algorithm
3. Implementation
4. Optimization
5. Conclusion
1. Introduction

- Convert a 3D model to 2D bitmap

- 3D objects are represented with geometric primitives
  -- triangles (triangular mesh)

- Three stages: projection, rasterization and z-culling

- A large amount of non-dependent operations
  -- Exploit Parallelism
1. Introduction
2. Algorithm - Projection

- Converts 3D triangles into 2D triangles.
- Orthographic projection. Perspective projection and rotation can be added.

- To project a 3D triangle onto x-y plane:
  1) Keep only x and y coordinates for all three vertices.
  2) Calculate the average of three z-coordinate values.
  3) Projection onto other planes are similar.
2. Algorithm - Projection
2. Algorithm - Rasterization

- Transforms 2D triangles into pixels.

- Barycentric Coordinates
  1) Determine the bounding box of the triangle by finding the Min and Max x and y values of the three vertexes
  2) Iterate over each point of the rectangular boundary and paint the pixel if it's inside the triangle
2. Algorithm - Rasterization
2. Algorithm – Z-culling

- Removes hidden surfaces
- A z-buffer is used to remember the front-most pixel
3. Implementation – Algorithm

- Dataflow

```
3D Triangle
  ↓
Projection
  ↓
2D Triangle
  ↓
Rasterization
  ↓
Z-culling
  ↓
  pixels
  ↓
FrameBuffer
  ↓
Resolution: 256x256
```
3. Implementation - Interface

- HLS stream
  - FIFO of infinite depth
  - Sequential read & write
- Sending Data
  - 3D triangle: \( x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2 \)
  - 8bit value (9) – 32 bit rendering-out (3)
- Receiving Data
  - Frame buffer: 256x256 pixels
  - 8bit value (64K) – 32bit rendering-in (16K)
4. Optimization

4.1 pipeline – 1.33x

- **Rasterization**: loop for searching pixels in the given triangle
  - Target II: 1, Final II: 1, Depth: 3.
  - Loop trip count: 1 ~ 924

- **Z-culling**: loop for updating z-buffer and output
  - Target II: 1, Final II: 2, Depth: 4.
  - Loop trip count: 0 ~ 223

- **Framebuffer**: loop for updating pixels
  - Target II: 1, Final II: 1, Depth: 2.
  - Loop trip count: 0 ~ 223
4. Optimization

4.2 Dataflow

(A) Without Dataflow Pipelining


(B) With Dataflow Pipelining


void top (a, b, c, d) {
  ...
  func_A(a, b, 11);
  func_B(c, 11, 12);
  func_C(12, d);
  return d;
}
4. Optimization

4.2 dataflow

Predicted speedup: $\frac{180}{89} \approx 2x$

However, real speedup: $1.02x$

Overall speedup: $1.37x$
5. conclusion

• 3D rendering software system
  ➢ Projection
  ➢ Rasterization
  ➢ Z-culling
  ➢ FrameBuffer

• 3D rendering hardware system
  ➢ Fixed-point variable (ap_int)

• Software-hardware interface
  ➢ HLS stream: infinite FIFO
  ➢ Rendering-in: 3D triangle mesh
  ➢ Rendering-out: FrameBuffer 256x256

• Speedup: **120x**
  ➢ Hardware : 90x
  ➢ Pipeline + Dataflow : 30x

• Resource usage

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM</td>
<td>78 (27%)</td>
</tr>
<tr>
<td>DSP</td>
<td>21 (9%)</td>
</tr>
<tr>
<td>FF</td>
<td>1236 (1%)</td>
</tr>
<tr>
<td>LUT</td>
<td>1698 (3%)</td>
</tr>
</tbody>
</table>