Building a Next-Generation Cloud Gaming Platform with Planar Map Streaming and Distributed Rendering

Pin Chun Wang
May, 2017
Outline

1. Introduction
2. Planar Map
3. System Architecture
4. Planar Map Compression
5. Evaluations
6. Future Work and Conclusion
Why Cloud Gaming

For gamers,

○ require constantly updating hardware
○ are locked to a specific computer

For developers,

○ need to solve hardware compatible issues
○ need to support multiple platforms
Cloud Gaming Platforms

- Perform game logic execution on server
  → thin client is enough
- Gaming result is streamed to client
  → supports clients on multiple platforms
Limitations of Existing Solutions

Existing Solutions

• PlayStation

• Gaming Anywhere

Limitations

• High bandwidth consumption
• Limited scalability
• Little room for optimizations

Gaming Anywhere: http://gaminganywhere.org/doc.html
Classifications of Cloud Gaming Platforms

Contributions

• We propose a **distributed rendering server-client framework** for cloud gaming platforms.

• We optimize the proposed system by exploring and compressing a new datatype, named planar map [1].

Outline

1. Introduction

2. Planar Map

3. System Architecture

4. Planar Map Compression

5. Evaluations

6. Future Work and Conclusion
Planar Map Rendering

- Planar map is a sequence of visible triangles
- 3D mesh rendering pipeline draws everything in 3D world while 2D planar map rendering pipeline only renders visible triangles
- Ellis et al. [1] proposed a real-time planar map pipeline

Silhouette Detection

- Silhouette is an edge shared by a frontfacing and a backfacing triangle
- We use a hash table to record all the edges in 3D scene as entry
- We check whether the corresponding faces’ normal are opposite signs in the z-axis
1. Project the triangle and silhouette onto the view plane
2. Clip each triangle against its list of overlapping silhouette
3. Walks through the polygon’s vertices determine whether to: (i) output a polygon edge, (ii) generate a new vertex, (ii) output the clipped edge

Leverage the property that no triangle is partially occluded to remove triangles.
Discard the triangles whose centroid is overlapped by any other triangle.
Then, we obtain a set of triangles with the depth complexity of one.
Outline

1. Introduction
2. Planar Map
3. System Architecture
4. Planar Map Compression
5. Evaluations
6. Future Work and Conclusion
Planar Map Format

In every planar map, we need to describe:

- Viewing matrix
- Draw call number and size
- 2D triangle description, including geometric information, texture coordinates, and normals
Coordinate Systems

Cartesian Coordinates,

- Pros: preserve spatial property within triangles
- Cons: no range limitation, require 30 entries for a triangle

Barycentric Coordinates, describe vertex information, within a triangle

- Pros: shorter indexes for triangles, common triangle patterns on unclipped triangles
- Cons: correlation among vertices harder to be leveraged
Coordinate Systems

Cartesian Coordinates,

- Pros: preserve the spatial property within triangles
- Cons: no range limitation, require 30 entry for a triangle

Barycentric Coordinates, describe vertex information, within a triangle

- Pros: shorter indexes for triangles, common triangle patterns on unclipped triangles
- Cons: correlation among vertices is harder to be leveraged
Outline

1. Introduction
2. Planar Map
3. System Architecture
4. Planar Map Compression
5. Evaluation
6. Future Work and Conclusion
Mesh Compression Pipeline

- We record three scenes in 720p resolution, in which we vary the number of the popular Bunny model, among 1, 2, and 8.
- We evaluate the performance by video quality metrics and compression ratio.

Quantization

- **Uniform**: divide the representative range in equal size
- **Scale**: apply Lloyd’s algorithm on individual dimensions sequentially
- **Vector**: all dimensions are jointly quantized using K-means

Decision: Uniform Quantization

Uniform quantization outperforms other modules since barycentric coordinates range between 0~1.
Delta Prediction

To leverage the property that close-by vertices share similar information, a prediction algorithm may use previous coordinates to predict current coordinates.

Decision: no delta prediction

Barycentric coordinates do not share spacial property
Therefore, delta prediction harms its compression ratio.
Entropy Encoding

- Huffman: builds a tree with the more frequent elements at lower levels of the tree
- Arithmetic: converts symbol sequence into a floating point between 1 and 0, and shrinks the interval based on the symbol probability
- LZMA: a dictionary compressor, which encodes a stream with an adaptive binary range coder

Decision: LZMA encoding
Compression Pipeline (Summary)

- Quantization
  - Vector
  - Scale
  - Uniform

- Delta Prediction

- Entropy
  - Huffman
  - Arithmetic
  - LZMA

- Uniform Quantization

- Delta prediction

- LZMA
Outline

1. Introduction
2. Planar Map
3. System Architecture
4. Planar Map Compression
5. Evaluations
6. Future Work and Conclusion
System Architecture (Revised)
Setup

Environment:

• i7 3.4 GHz
• GPU: NVidia Quadro M4000

Baseline:

• image-based solution
• state-of-art x265 video codec

Scenes:

• diverse model complexity : { Basic, Fine Grained }
• numbers of bunny : { 1, 4, 8 }
• camera speed : { slow, fast }
Bandwidth Consumption

- We vary compression depths in compression pipeline
- Bandwidth consumption ranges from 1.5 to 4 Mbps for basic bunnies, and 2.5 to 17 for fine-grained bunnies
Video Quality (PSNR)

- Up to 5 dB improvement is possible, and as long as the bit-depth is $\geq 7$ bits, our proposed solution results in higher PSNR in basic bunnies scenes.
Video Quality (SSIM)

- Distributed rendering preserves image structures in low bitrate situation
- The gap is as high as 0.14 in SSIM

Outperforms x265 in SSIM under low bitrate scanario
Ultra-High Resolution Display

- Our proposed solution scales well to high resolution applications, such as 360 videos and VR

Our solution scales well to high resolution applications

PSNR = 25

PSNR = 30
Running Time

- Run computation demanding process on server side (< 1%)
- Achieve real-timeness (basic scene)

<table>
<thead>
<tr>
<th></th>
<th>Server side (mean / max) in ms.</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection</td>
<td>0.27 / 0.28</td>
<td>0.22 / 0.48</td>
</tr>
<tr>
<td>Clipping</td>
<td>25.56 / 33.41</td>
<td>60.55 / 88.14</td>
</tr>
<tr>
<td>Occlusion</td>
<td>1.94 / 2.68</td>
<td>17.43 / 24.02</td>
</tr>
<tr>
<td>Rendering</td>
<td>0.83 / 3.13</td>
<td>= 28 ms</td>
</tr>
</tbody>
</table>

Basic scene

Fine-grained scene
Outline

1. Introduction
2. Planar Map
3. System Architecture
4. Planar Map Compression
5. Evaluations

6. Future Work and Conclusion
Discussion

• PEVQ is an metrics of Perceptual Evaluation of Video Quality (PEVQ) [1] described in ITU-T J.247 Annex B

The curve of our solution is fairly flat compared to x265

Sample Video
Sample Video

X265

OPVQ: 3.43

Our Solution

OPVQ: 1.54
Hole Filling

- When zooming in the rendered scene, we find there are small holes within vertices, which may lead to lower PEVQ scores.
Future Work

• Filling holes by morphological antialiasing
• Integrate with game engine
• Evaluate system performance with user study

Conclusion

• We proposed a cloud gaming platforms with planar map
• Compared to video streaming based platforms, our planar map based platform:
  o achieves higher video quality at the same bitrate
  o runs fast, especially at the client side
  o scales well to ultra-high-resolution displays