m56732
Evaluation on Trisoup vertex sorting

Kyohei Unno, Kei Kawamura

KDDI Corp. (KDDI Research, Inc.)

Overview

KDDI Research

Background

- Vertex sorting in Trisoup was done by an approximation of arctangent.
- However, the approximation had a bug and bugfix proposed in m55951 was adopted to the test model at the previous meeting.
- On the other hand, G-PCC has another “high precision” approximation of arctangent (iatan2).

Methods

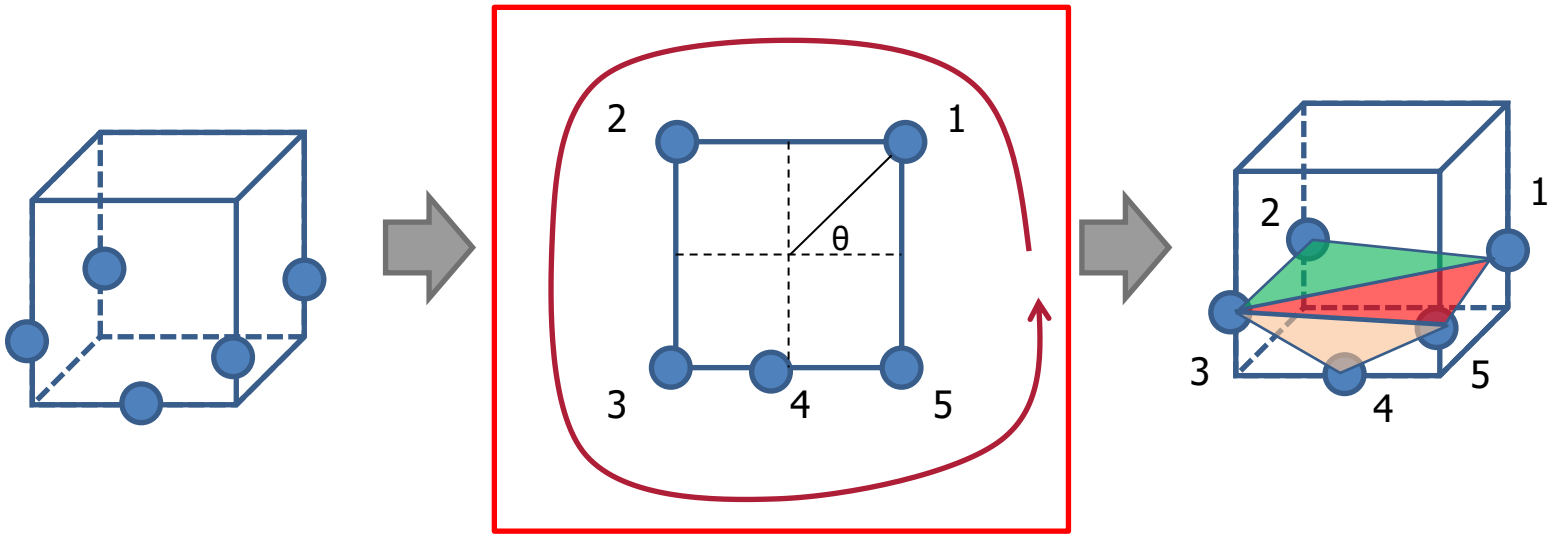
- iatan2 : Sorting vertices from 9 o’clock with anticlockwise rotation.
- TMC13 v13 (m55951) : Originally sorting vertices from 3 o’clock with anticlockwise rotation. This process is much simpler than iatan2.

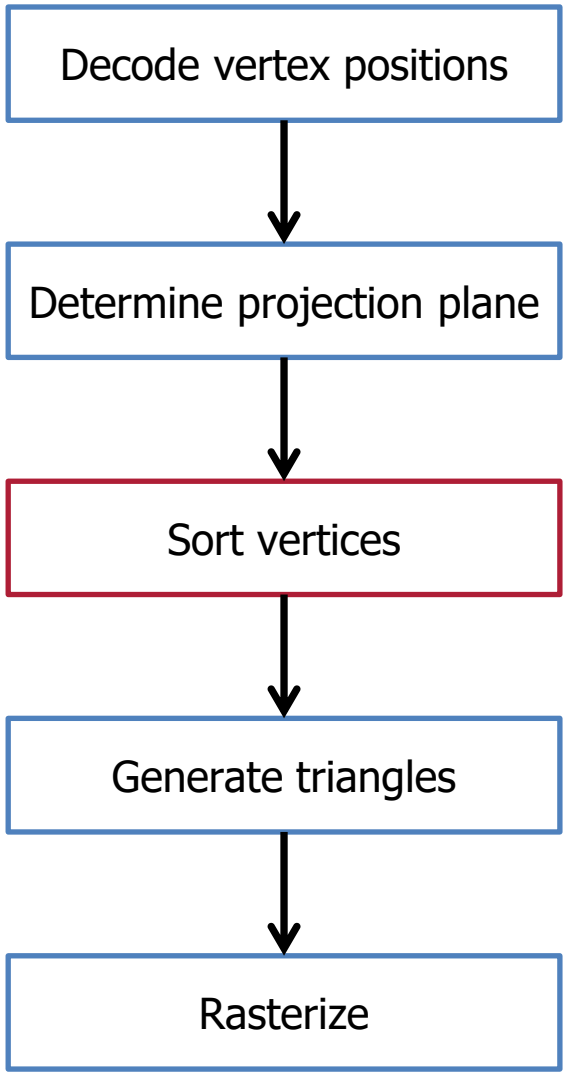
Experimental results

- m55951(from 3 o’clock) vs iatan2 : BD-rates for geometry are -0.1%, -0.3% (D1, D2).
- m55951(from 9 o’clock) vs iatan2 : identical results.

Vertex sorting in Trisoup

- In the decoding process of Trisoup, vertices of each Trisoup node are sorted to generate triangles.
- Previously, an approximation of arctangent was used to calculate θ for sorting.
- However, the approximation had a bug and bugfix proposed in m55951 was adopted to the test model at the previous meeting.





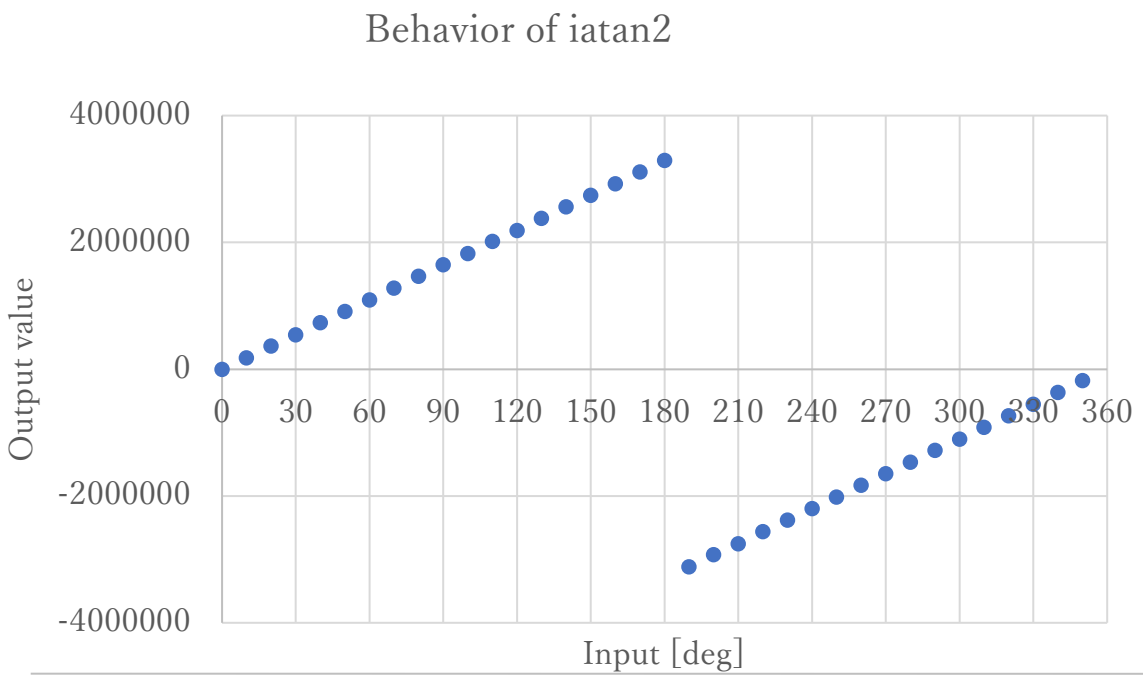
Another “high precision” approximation of arctangent

KDDI Research

■ iatan2()

- Vertices can be sorted from 9 o’clock with anticlockwise rotation.
- Worst case complexity for a vertex

ADD	MUL	DIV	SHIFT	COMP
24	8	0	26	8

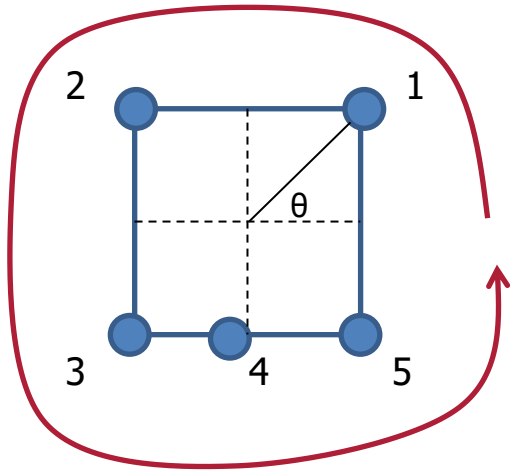


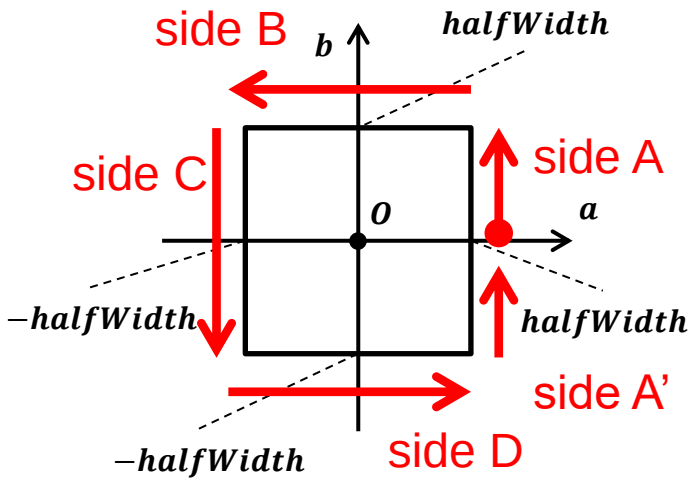

```
int
iatan2Core(int y, int x)
{
  using namespace atan2;

  // ratio y/x knowing that x,y >0 and y<=x
  if (x == 0)
    return 0;

  uint64_t rinv =
    irsqrt(uint64_t(x) * uint64_t(x) + uint64_t(y) * uint64_t(y));
  int r = (y * rinv) >> 20;  // 40 - 20 = 20 bits precision
  int idx = r >> 11;
  int lambda = r - (idx << 11);
  return kAsin[idx] + (lambda * (kAsin[idx + 1] - kAsin[idx]) >> 11);
}
int
iatan2(int y, int x)
{
  int xa = std::abs(x);
  int ya = std::abs(y);

  // atan or pi/2 - atan
  int t = ya <= xa ? iatan2Core(ya, xa) : 1647099 - iatan2Core(xa, ya);
  if (x < 0)
    t = 3294199 - t;  // pi -atan

  return y < 0 ? -t : t;
}
```

Vertex sorting in TMC13 v13 (m55951)

- **Vertices are always on edges of the projected square.**
  - Coordinate values are directly used to sort instead of approximated Atan.
  - Ex) When a vertex is on the horizontal edge, coordinate value of vertical axis is used.
  - A offset value is added to ensure circulate order.
  - Worst case complexity for a vertex.

ADD	MUL	DIV	SHIFT	COMP
1	1	0	0	3



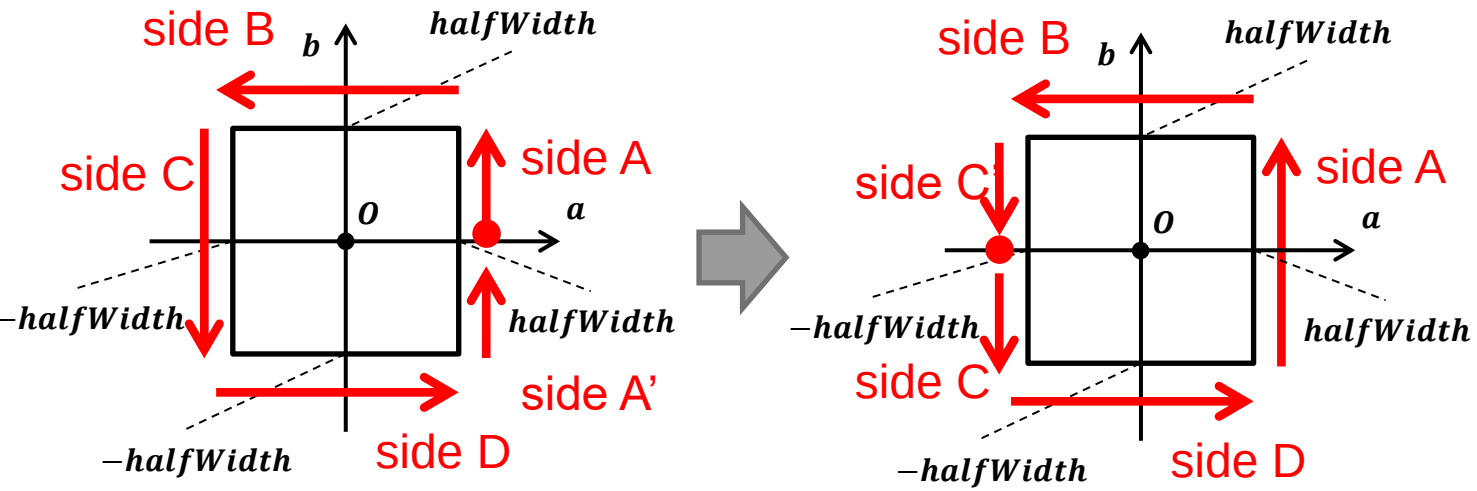



```
int32_t
trisoupVertexScore(int32_t x, int32_t y, int32_t halfWidth)
{
  int32_t score;

  if(x >= halfWidth){
    score = y; // for side A
    if (y < 0) {
      score += (halfWidth * 8); // for side A'
    }
  } else if (y >= halfWidth) {
    score = -x + (halfWidth * 2); // for side B
  } else if (x <= -halfWidth) {
    score = -y + (halfWidth * 4); // for side C
  } else {
    score = x + (halfWidth * 6); // for side D
  }

  return score;
}
```

Starting point for sorting

■ Starting point for sorting

- iatan2 : 9 o’clock
- TMC13 v13 (m55951) : originally 3 o’clock.
  Starting point can be changed (e.g. from 9 o’clock) easily.
  Complexity does not change.




```
int32_t
trisoupVertexScore(int32_t x, int32_t y, int32_t halfWidth)
{
  int32_t score;

  if(x >= halfWidth){
    score = y + (halfWidth * 4); // for side A
  } else if (y >= halfWidth) {
    score = -x + (halfWidth * 6); // for side B
  } else if (x <= -halfWidth) {
    score = -y; // for side C
    if (y >= 0) {
      score += (halfWidth * 8); // for side C’
    }
  } else {
    score = x + (halfWidth * 2); // for side D
  }

  return score;
}
```

Experimental results (1/2)

■ Conditions

- Anchor : TMC13-v12.0 + m55951 (from 3 o'clock)
- Test : TMC13-v12.0 + iatan2 (from 9 o'clock)
- Trisoup – RAHT (Only C2 condition, Cat1 Sequences)

■ Objective results

- BD-rates for geometry are -0.1%, -0.3% (D1, D2).
- Coding performance difference is very minor.

C2_ai	lossy geometry, lossy attributes [all intra]					
	End-to-End BD-AttrRate [%]				Geom. BD-TotGeomRate [%]	
	Luma	Chroma Cb	Chroma Cr	Reflectance	D1	D2
Cat1-A average	0.3%	0.0%	0.4%		-0.1%	-0.6%
Cat1-B average	-0.3%	0.1%	0.5%		-0.1%	0.0%
Cat3-fused average	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Cat3-frame average				#DIV/0!	#DIV/0!	#DIV/0!
Overall average	0.0%	0.0%	0.5%	#DIV/0!	-0.1%	-0.3%
Avg. Enc Time [%]	104%					
Avg. Dec Time [%]	95%					

Experimental results (2/2)

KDDI Research

■ Conditions

- Anchor : TMC13-v12.0 + m55951 (from 9 o'clock)
- Test : TMC13-v12.0 + iatan2 (from 9 o'clock)
- Trisoup – RAHT (Only C2 condition, Cat1 Sequences)

■ Objective results

- Coding performances are identical.
- Difference in the previous slide just come from difference of the starting point.

C2_ai	lossy geometry, lossy attributes [all intra] End-to-End BD-AttrRate [%]				Geom. BD-TotGeomRate [%]	
	Luma	Chroma Cb	Chroma Cr	Reflectance	D1	D2
Cat1-A average	0.0%	0.0%	0.0%		0.0%	0.0%
Cat1-B average	0.0%	0.0%	0.0%		0.0%	0.0%
Cat3-fused average	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
Cat3-frame average				#DIV/0!	#DIV/0!	#DIV/0!
Overall average	0.0%	0.0%	0.0%	#DIV/0!	0.0%	0.0%
Avg. Enc Time [%]	100%					
Avg. Dec Time [%]	100%					

Conclusion

KDDI Research

- **Background**
  - Vertex sorting in Trisoup was done by an approximation of arctangent.
  - However, the approximation had a bug and bugfix proposed in m55951 was adopted to the test model at the previous meeting.
  - On the other hand, G-PCC has another “high precision” approximation of arctangent (iatan2).
- **Methods**
  - iatan2 : Sorting vertices from 9 o’clock with anticlockwise rotation.
  - TMC13 v13 (m55951) : Originally sorting vertices from 3 o’clock with anticlockwise rotation. This process is much simpler than iatan2.
- **Experimental results**
  - m55951(from 3 o’clock) vs iatan2 : BD-rates for geometry are -0.1%, -0.3% (D1, D2).
  - m55951(from 9 o’clock) vs iatan2 : identical results.
- **Recommendation**
  - No action (keep the current TMC13 v13 implementation).