m55951
Bugfix of vertices sorting in Trisoup by alternative method

Kyohei Unno, Kei Kawamura

KDDI Corp. (KDDI Research, Inc.)

Overview

KDDI Research

- **Problem statement**
  - In the previous meeting, a problem on the subjective quality of Trisoup was pointed out.
  - We found that the main cause of the problem seems to be in the vertices sorting process by approximated Atan.

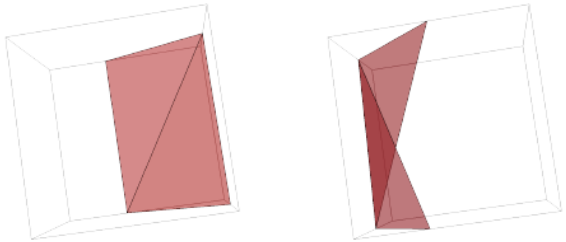
- **Proposal**
  - An alternative vertex sorting process is proposed.
  - Vertices are sorted by their coordinate value directly instead of using approximated Atan.
  - The proposed method solves the problem and simplifies the process (division free).

- **Experimental results**
  - Almost all of the subjective problems are disappeared.
  - Some coding losses observed. However, they are identical results in case of using double precision Atan (not approximated).

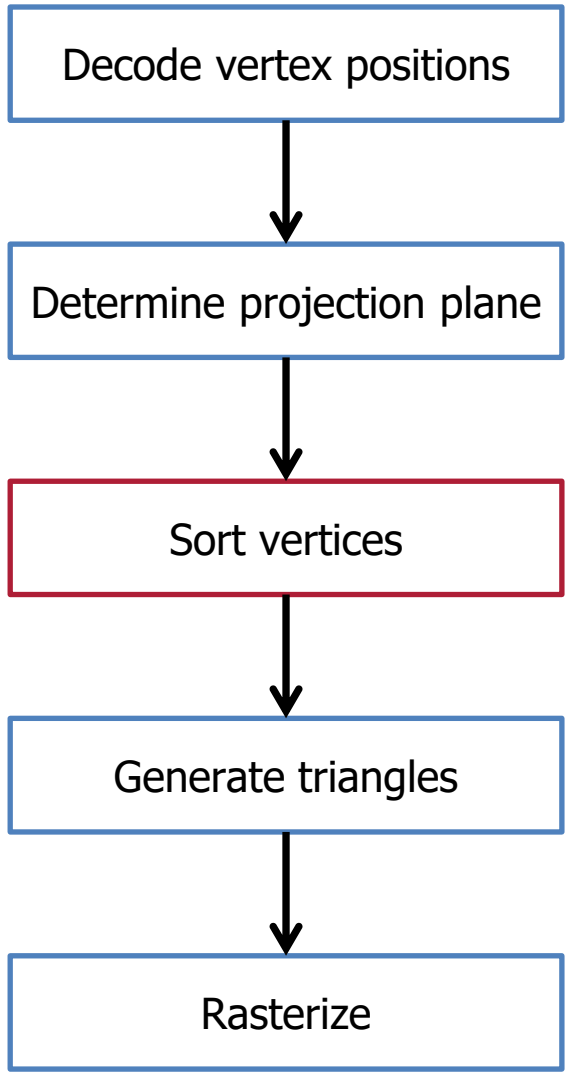
Problem Statement

- **In the previous meeting, a problem on the subjective quality of Trisoup was pointed out in m55493.**
  - **It is reported that some malformed nodes are observed.**
- **We found that the main cause of generating malformed nodes seems to be in the vertices sorting process**

(a) Reconstructed (b) Original

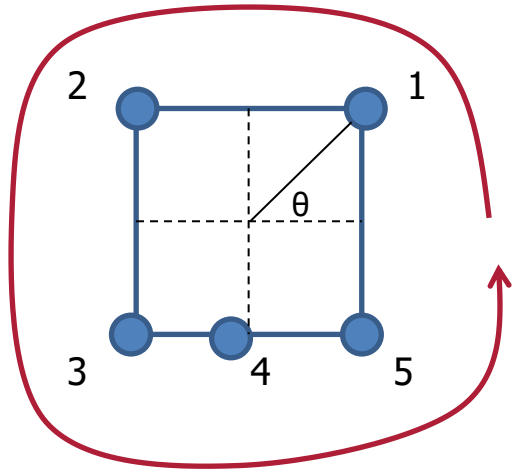


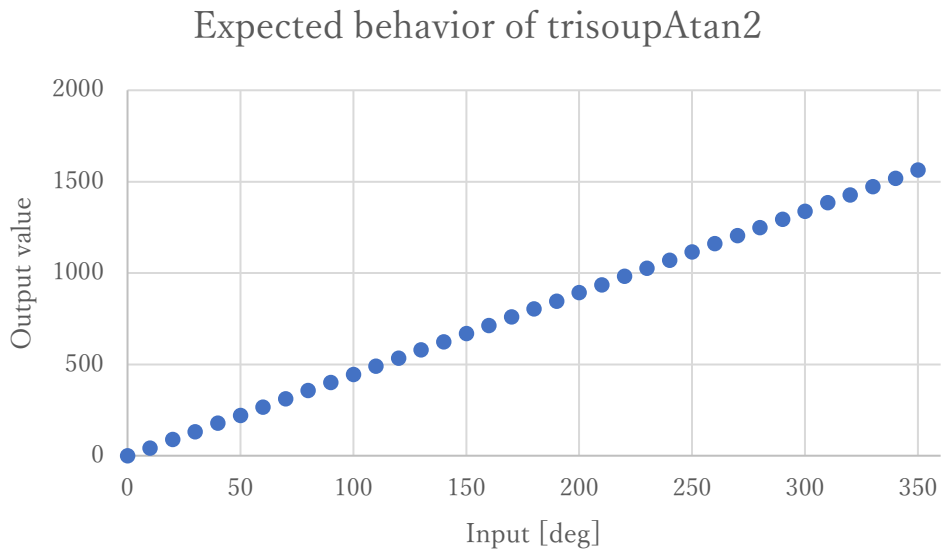
(a) Well-formed node (b) Malformed node

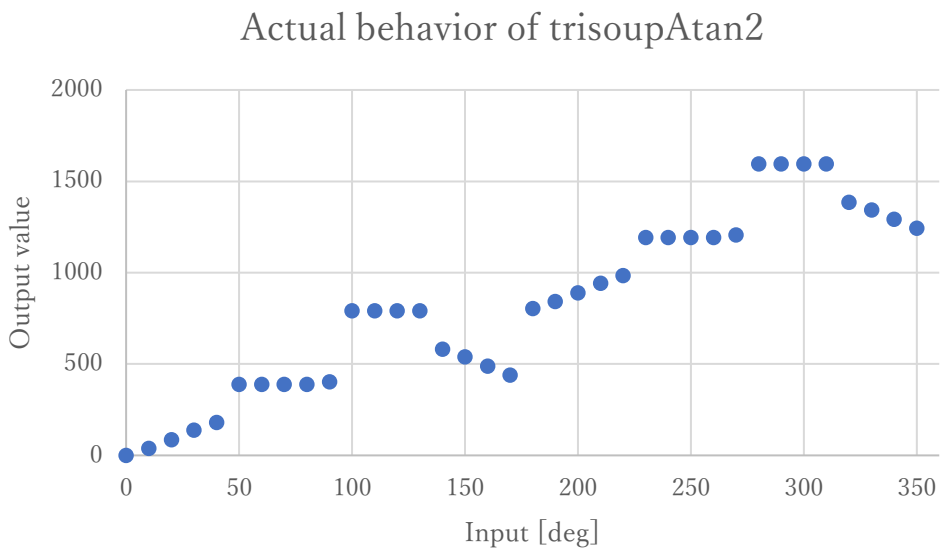


Problem Statement (Cont.)

- **The current sorting process**
  - Angle of each vertex is calculated by trisoupAtan2().
  - trisoupAtan2() is an approximation of arctangent.
  - Vertices are sorted by angles in ascending order.
- **Expected and actual outputs trisoupAtan2().**
  - Actual outputs lead to wrong ordering.







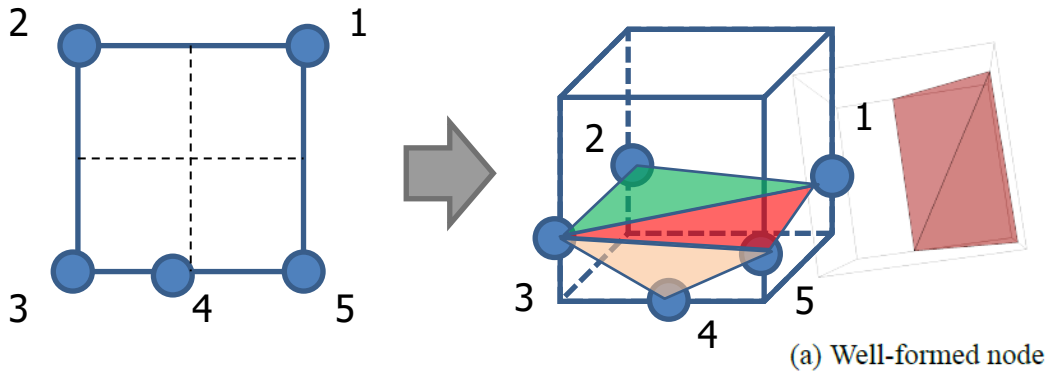
Problem Statement (Cont.)

■ **Relationship between vertex sorting results and triangle is hardcoded.**

n	triangles
3	(1,2,3)
4	(1,2,3), (3,4,1)
5	(1,2,3), (3,4,5), (5,1,3)
6	(1,2,3), (3,4,5), (5,6,1), (1,3,5)
7	(1,2,3), (3,4,5), (5,6,7), (7,1,3), (3,5,7)
8	(1,2,3), (3,4,5), (5,6,7), (7,8,1), (1,3,5), (5,7,1)
9	(1,2,3), (3,4,5), (5,6,7), (7,8,9), (9,1,3), (3,5,7), (7,9,3)
10	(1,2,3), (3,4,5), (5,6,7), (7,8,9), (9,10,1), (1,3,5), (5,7,9), (9,1,5)
11	(1,2,3), (3,4,5), (5,6,7), (7,8,9), (9,10,11), (11,1,3), (3,5,7), (7,9,11), (11,3,7)
12	(1,2,3), (3,4,5), (5,6,7), (7,8,9), (9,10,11), (11,12,1), (1,3,5), (5,7,9), (9,11,1), (1,5,9)

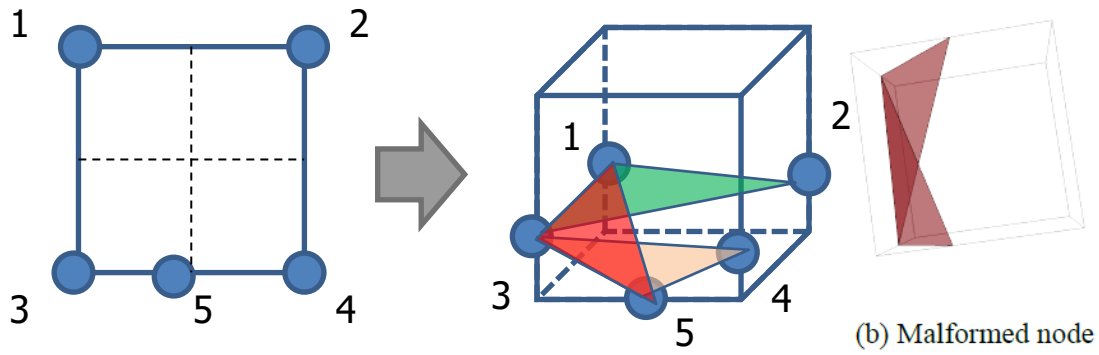
■ **Therefore, wrong ordering may cause malformed nodes.**

Correct order ☺



(a) Well-formed node

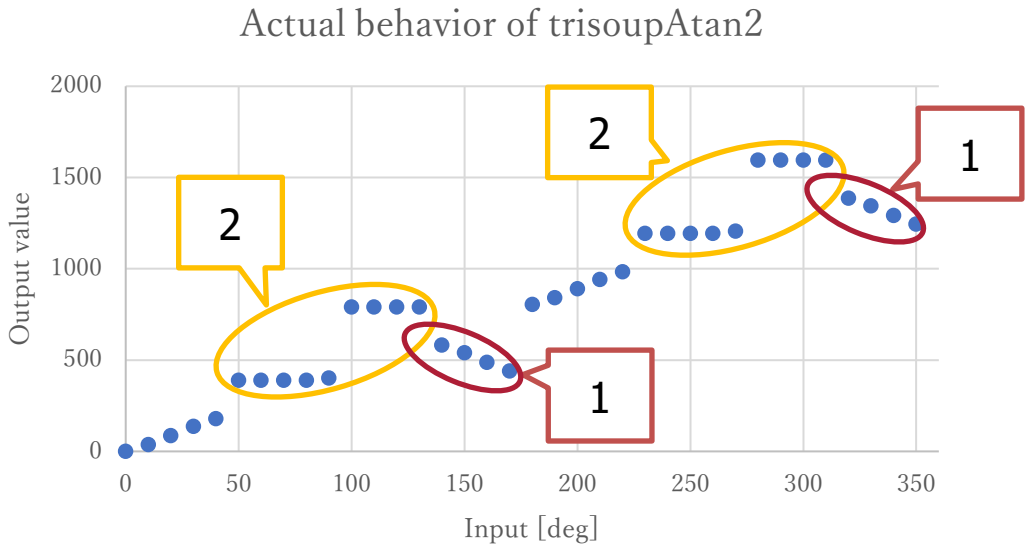
Wrong order ☹



(b) Malformed node

Problem Statement (Cont.)

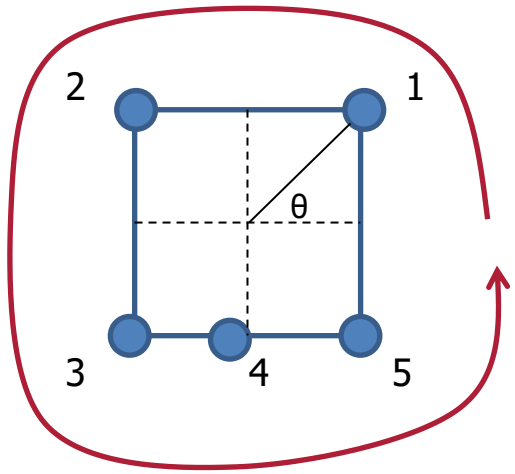
- **There are two problems.**
  1. **Rad is always calculated in the first quadrant.**
     - Order is reversed in the second and fourth quadrant
  2. **In else part, idx is constant.**
     - Resulted atan is also constant.
- **Additionally, divisions are used.**

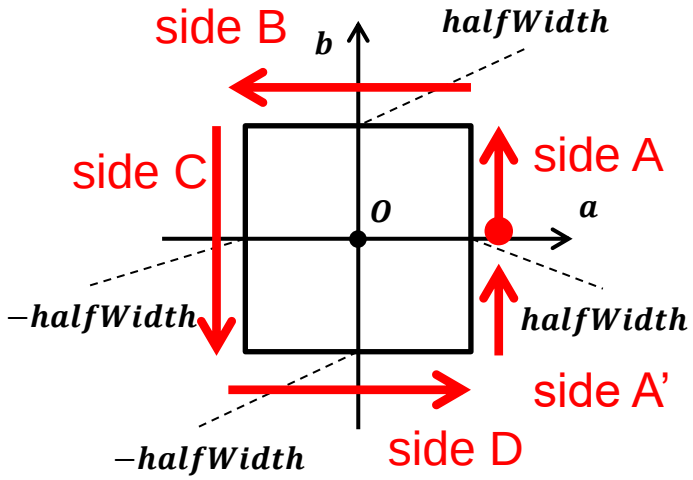



```
int32_t
trisoupAtan2(int32_t x, int32_t y)
{
  assert(x != 0 && y != 0);
  if (y == 0) {
    if (x < 0)
      return 804;  // PI * (1<< kTrisoupFpBits)
    else
      return 0;
  } else if (x == 0) {
    if (y > 0)
      return 402;  // (PI/2)   * (1<< kTrisoupFpBits)
    else
      return 1206;  // (PI*3/2) * (1<< kTrisoupFpBits)
  } else {
    int idx = 0;
    int z = abs((y << 8) / x);  //rad is calc in (x>0 && y>0) domain
    if (z <= 256) {             //1<<kTrisoupFpBits
      idx = z / 12;             //0.05<<kTrisoupFpBits
    } else {
      idx = z > 40 ? 40 : z;
    }

    static const int kAtanLut[41] = {
      0,   12,  25,  38,  50,  62,  74,  86,  97,  108, 118, 128, 138, 147,
      156, 164, 172, 180, 187, 194, 201, 283, 319, 339, 351, 359, 365, 370,
      373, 376, 378, 380, 382, 383, 385, 386, 387, 387, 388, 389, 389};
    int atan = kAtanLut[idx];

    //offset
    if (x < 0 && y > 0)
      atan += 402;  // + PI/2
    else if (x < 0 && y < 0)
      atan += 804;  // + PI
    else if (x > 0 && y < 0)
      atan += 1206;  // + PI*3/2
    return atan;
  }
}
```

Proposal

- It is proposed that an alternative vertex sorting.
  - It solves the problem with simpler method than the current implementation.
- 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.






```
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;
}
```

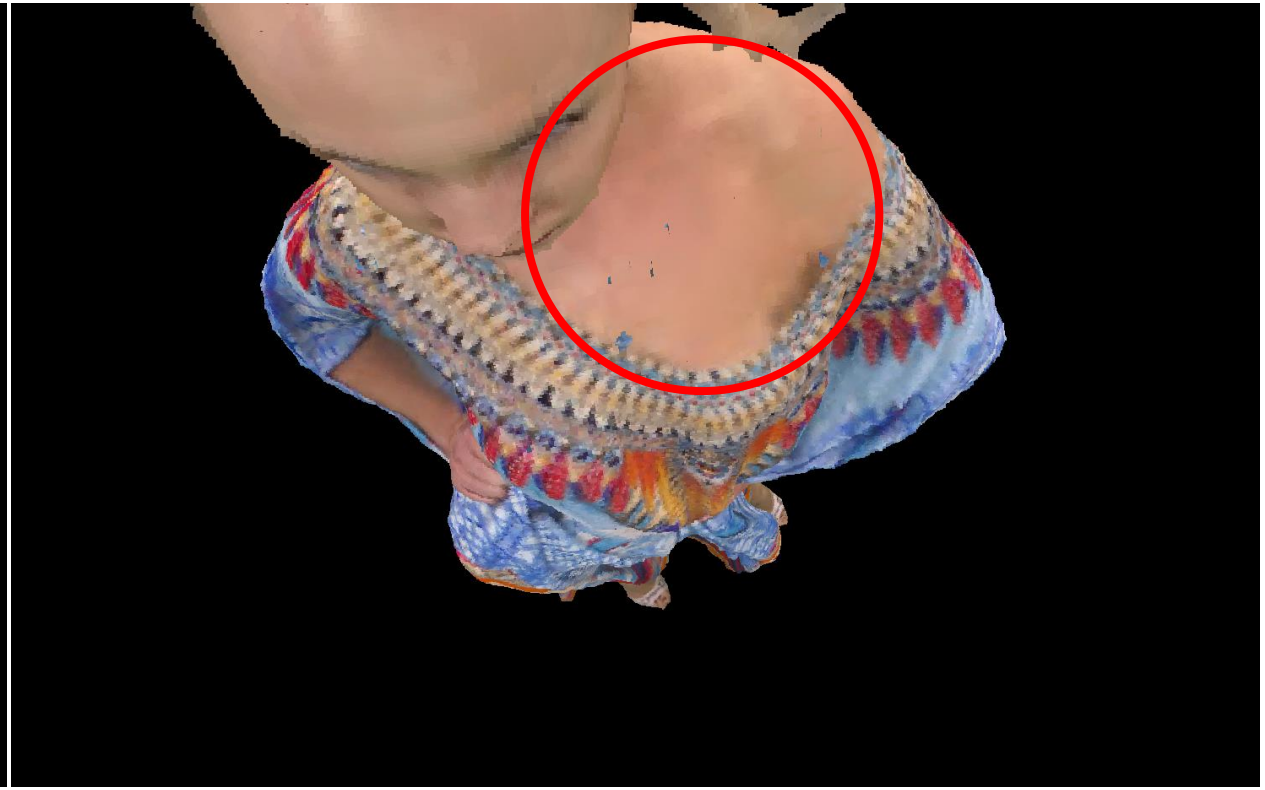
Experimental results

Subjective evaluation

- **Almost all of the subjective problems (small holes) are disappeared.**

Reconstructed point clouds (longdress_vox10_1300 r03)

Anchor (TMC13-v12.0)



Proposed method

Experimental results

■ Subjective evaluation

- **Almost all of the subjective problems (small holes) are disappeared.**

Reconstructed point clouds (queen_0200 r03)

Anchor (TMC13-v12.0)

Proposed method

Experimental results

KDDI Research

■ Conditions

- Anchor : TMC13-v12.0
- Trisoup – RAHT (Only C2 condition, Cat1 Sequences)

■ Objective results

- BD-rates for geometry are 0.4%, 2.1% (D1, D2).
- This result is identical to the case of using double precision Atan (std::atan2()).

```
int32_t
trisoupAtan2(int32_t x, int32_t y)
{
    int atan;
    double PI = 3.141592;
    if (y < 0){
        atan = (402 * (std::atan2(y, x) + 2 * PI) * 2 / PI);
    }else{
        atan = (402 * std::atan2(y, x) * 2 / PI);
    }
    return atan;
}
```

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	-1.0%	-0.5%	-0.3%		-0.4%	1.2%
Cat1-B average	0.3%	-0.7%	-0.3%		1.1%	2.9%
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	-0.3%	-0.6%	-0.3%	#DIV/0!	0.4%	2.1%
Avg. Enc Time [%]	100%					
Avg. Dec Time [%]	98%					

Experimental results

- **The peak loss is observed at palazzo_carignano_dense_vox14 (about 10%).**
- **On the other hand, subjective quality is improved.**

Reconstructed point clouds (palazzo_carignano_dense_vox14 r03)

Anchor (TMC13-v12.0)

Proposed method

Conclusion

- **Problem statement**
  - In the previous meeting, a problem on the subjective quality of Trisoup was pointed out.
  - We found that the main cause of the problem seems to be in the vertices sorting process by approximated Atan.
- **Proposal**
  - An alternative vertex sorting process is proposed.
  - Vertices are sorted by their coordinate value directly instead of using approximated Atan.
  - The proposed method solves the problem and simplifies the process (division free).
- **Experimental results**
  - Almost all of the subjective problems are disappeared.
  - Some coding losses observed. However, they are identical results in case of using double precision Atan (not approximated).
- **Recommendation**
  - The proposal is adopted to the next version of the test model as the starting point of solving the problem.