Referências Bibliográficas

[1] BÆRENTZEN, J.; AANÆS, H.. **Signed distance computation using the angle weighted pseudo-normal**. IEEE Transactions on Visualization and Computer Graphics, 11(3):243–253, may 2005.

[2] BÆRENTZEN, J. A.. **Manipulation of volumetric solids with applications to sculpting**. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002. Supervised by Niels Jørgen Christensen, IMM.

[3] BAJAJ, C.; BERNARDINI, F. ; XU, G.. **Adaptive reconstruction of surfaces and scalar fields from dense scattered trivariate data**. Computer Science Technical Report TR 95-028, Purdue University, 1995.

[4] COHEN-OR, D.; SOLOMOVIC, A. ; LEVIN, D.. **Three-dimensional distance field metamorphosis**. ACM Trans. Graph., 17(2):116–141, 1998.

[5] CORNEA, N. D.; SILVER, D. ; MIN, P.. **Curve-skeleton properties, applications, and algorithms**. IEEE Transactions on Visualization and Computer Graphics, 13(3):530–548, 2007.

[6] DE FIGUEIREDO, L. H.; VELHO, L. ; OLIVEIRA, J. B. D.. **Revisiting adaptively sampled distance fields**. In: SIBGRAPI '01: PROCEEDINGS OF THE XIV BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING (SIBGRAPI'01), p. 377, Washington, DC, USA, 2001. IEEE Computer Society.

[7] DEY, T. K.. **Curve and Surface Reconstruction: Algorithms with Mathematical Analysis (Cambridge Monographs on Applied and Computational Mathematics)**. Cambridge University Press, New York, NY, USA, 2006.

[8] FRISKEN, S. F.; PERRY, R. N.. **A computationally efficient framework for modeling soft body impact**. Technical Report TR2001-011, Mitsubishi Electric Research Laboratories, 2001.

[9] FRISKEN, S. F.; PERRY, R. N.. **Designing with distance fields**. Technical Report TR2006-054, Mitsubishi Electric Research Laboratories, 2006.

[10] FRISKEN, S. F.; PERRY, R. N.; ROCKWOOD, A. P. ; JONES, T. R.. **Adaptively sampled distance fields: a general representation of shape for computer graphics**. In: SIGGRAPH '00: PROCEEDINGS OF THE 27TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, p. 249–254, New York, NY, USA, 2000. ACM Press.

[11] HABLE, J.; ROSSIGNAC, J.. **Blister: Gpu-based rendering of boolean combinations of free-form triangulated shapes**. In: SIGGRAPH '05: ACM SIGGRAPH 2005 PAPERS, p. 1024–1031, New York, NY, USA, 2005. ACM.

[12] HADWIGER, M.; SIGG, C.; SCHARSACH, H.; BÜHLER, K. ; GROSS, M.. **Real-time ray-casting and advanced shading of discrete isosurfaces**. In: PROCEEDINGS OF EUROGRAPHICS 2005, p. 303–312, 2005.

[13] HART, J. C.. **Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces**. The Visual Computer, 12(10):527–545, 1996.

[14] HARTMANN, E.. **On the curvature of curves and surfaces defined by normalforms**. Comput. Aided Geom. Des., 16(5):355–376, 1999.

[15] HORN, D. R.; SUGERMAN, J.; HOUSTON, M. ; HANRAHAN, P.. **Interactive k-d tree gpu raytracing**. In: I3D '07: PROCEEDINGS OF THE 2007 SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, p. 167–174, New York, NY, USA, 2007. ACM.

[16] JOHNSON, D.; COHEN, E.. **Minimum distance queries for polygonal and parametric models.** Technical Report UUCS-97-003, 31, 1997.

[17] JONES, M. W.; BAERENTZEN, J. A. ; SRAMEK, M.. **3D Distance Fields: A Survey of Techniques and Applications**. IEEE Transactions on Visualization and Computer Graphics, 12(4):581–599, 2006.

[18] KAPASI, U. J.; RIXNER, S.; DALLY, W. J.; KHAILANY, B.; AHN, J. H.; MATTSON, P. ; OWENS, J. D.. **Programmable stream processors**. Computer, 36(8):54–62, 2003.

[19] KAUFMAN, A.; COHEN, D. ; YAGEL, R.. **Volume graphics**. Computer, 26(7):51–64, 1993.

[20] LEFEBVRE, S.; HOPPE, H.. **Perfect spatial hashing**. In: SIGGRAPH '06: PROCEEDINGS OF THE 33TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, p. 579–588, New York, NY, USA, 2006. ACM Press.

[21] LOOP, C.; BLINN, J.. **Real-time GPU rendering of piecewise algebraic surfaces**. In: SIGGRAPH '06: ACM SIGGRAPH 2006 PAPERS, p. 664–670, New York, NY, USA, 2006. ACM Press.

[22] OWENS, J. D.; LUEBKE, D.; GOVINDARAJU, N.; HARRIS, M.; KRÜGER, J.; LEFOHN, A. E. ; PURCELL, T. J.. **A survey of general-purpose computation on graphics hardware**. In: EUROGRAPHICS 2005, STATE OF THE ART REPORTS, p. 21–51, aug 2005.

[23] PAYNE, B. A.; TOGA, A. W.. **Distance field manipulation of surface models**. IEEE Computer Graphics and Applications, 12(1):65–71, 1992.

[24] PERRY, R. N.; FRISKEN, S. F.. **Kizamu: a system for sculpting digital characters**. In: SIGGRAPH '01: PROCEEDINGS OF THE 28TH ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, p. 47–56, New York, NY, USA, 2001. ACM Press.

[25] PERRY, R. N.; FRISKEN, S. F.. **A new representation for device color gamuts**. Technical Report TR2001-009, Mitsubishi Electric Research Laboratories, 2001.

[26] POPE, J.; FRISKEN, S. F. ; PERRY, R. N.. **Dynamic meshing using adaptively sampled distance fields**. Technical Report TR2001-013, Mitsubishi Electric Research Laboratories, 2001.

[27] POPOV, S.; GÜNTHER, J.; SEIDEL, H.-P. ; SLUSALLEK, P.. **Stackless kd-tree traversal for high performance GPU ray tracing**. Computer Graphics Forum, 26(3):415–424, Sep 2007. (Proceedings of Eurographics).

[28] ROMEIRO, F.; VELHO, L. ; DE FIGUEIREDO, L. H.. **Hardware-assisted rendering of csg models**. sibgrapi, 0:139–146, 2006.

[29] SATHERLEY, R.; JONES, M. W.. **Vector-city vector distance transform**. In: COMPUTER VISION AND IMAGE UNDERSTANDING, volumen 82, p. 238–254, June 2001.

[30] SERRA, J.. **Image Analysis and Mathematical Morphology**. Academic Press, Inc., Orlando, FL, USA, 1983.

[31] SUD, A.; OTADUY, M. A. ; MANOCHA, D.. **Difi: Fast 3d distance field computation using graphics hardware**. In: COMPUTER GRAPHICS FORUM (PROCEEDINGS OF EUROGRAPHICS), volumen 3, p. 557–566, 2004.

[32] TOLEDO, R.; LEVY, B.. **Extending the graphic pipeline with new gpu-accelerated primitives**. In: INTERNATIONAL GOCAD MEETING, NANCY, FRANCE, 2004.

A
Código GLSL

Algoritmo A.1 Busca pela menor célula que contém **p** a partir do nível ℓ.

```
vec3 query( inout int level, const vec3 p ) {
  // check if a cell exists at the current 'level' and position 'p'
  vec3 k = S(level, p); // generate a key for the cell
  ivec2 cell = texture3D( cellTex, H(k) ).ra; // fetch cell
  int voxels = cell.y; // save the address of the cell's voxel block
  if( computeId(k) == cell.x ) { // is this the right cell?
    while( level <= maxLevel ) { // cell exists, try to descend the tree
      k = S( ++level, p ); // generate key for the child cell containing p
      cell = texture3D( cellTex, H(k) ).ra; // fetch cell
      if( computeId(k) != cell.x ) { // is this the right cell?
        --level; // no cell at this level, go back to the last valid level
        break;     // the last visited cell is the smallest one
      }
      voxels = cell.y; // keep track of the smallest cell
    }
  } else { // no cell at the current 'level' and position 'p'
    while( level > 0 ) { // ascend the octree until we find a valid cell
      k = S( --level, p ); // generate key for the parent cell
      cell = texture3D( cellTex, H(k) ).ra; // fetch cell
      if( computeId(k) == cell.x ) { // is this the right cell?
        voxels = cell.y; // the first cell we find upwards is the smallest
        break;             // so we are done
      }
    }
  } // map 1D address of the cell's voxel block to an unnormalized 3D texture coord
  return map1Dto3D( voxels, voxelTexDim ) + 0.5;
}
```

Algoritmo A.2 Lançador de raios baseado em traçado por esferas.

```
// Fragment Program:
uniform vec3  eye;          // ray origin in world space
uniform float isovalue;     // value of the isosurface; usually 0
uniform float tolerance;    // intersection tolerance for sphere-tracing
varying vec3  fragment_pos; // from the vertex program, in world space

float reconstructDist( const vec3 vb, const vec3 cp ) {
  return texture3D( voxelTex, ( vb + cp ) / voxelTexDim ).a;
}

void main()
{
  // ray starts at the fragment's position, on the surface of the ADF's AABB
  vec3 pos = fragment_pos; // current position along the ray
  vec3 dir = normalize( pos - eye ); // ray direction
  // compute the distances at which the ray enters and leaves the ADF's AABB
  float rayLength, maxRayLength;
  intersectAABB( eye, dir, rayLength, maxRayLength );
  // traverse the distance field until we hit the isosurface or leave the AABB
  int level = 1; // initial level for queries
  for( int i = 0; i < MAX_ITERATIONS; ++i ) {
    vec3 vb = query( level, pos ); // locate the cell's voxel block
    vec3 cp = toCellSpace( pos, level ); // pos into cell space
    float d = reconstructDist( vb, cp ) - isovalue;
    rayLength += d; // step along the ray
    pos = eye + dir * rayLength; // update the current position
    if( d < tolerance ) { // did we hit the isosurface?
      vec3 normal = reconstructNormal( vb, cp );
      gl_FragColor = shade( pos, normal, -dir );
      return;
    }
    if( rayLength >= maxRayLength ) // did we leave the AABB?
      discard; // ray does not intersect the isosurface
  }
  discard; // we have exceeded MAX_ITERATIONS
}
```