

# Supplementary material for ray/patch intersection from the Chapter 8 in **Ray Tracing Gems** book (<http://www.realtimerendering.com/raytracinggems>)

Copyright (c) 2019 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation and its licensors retain all intellectual property and proprietary rights in and to this software, related documentation and any modifications thereto. Any use, reproduction, disclosure or distribution of this software and related documentation without an express license agreement from NVIDIA Corporation is strictly prohibited.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED \*AS IS\* AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES

---

```

In[ ]:= (* This is a numerical example that corresponds to the accompanied C++ code;
see also Figure 8.4 *)

lerp[p0_, p3_, t_] := p0 * (1 - t) + p3 * t;
quadrilateral[u_, v_, vertices_List] := (1 - u) (1 - v) vertices[[1]] +
    u (1 - v) vertices[[2]] + u v vertices[[3]] + (1 - u) v vertices[[4]];

vertices = {{0, 0, 1}, {1, 0, 0.5}, {1.2, 1, 0.8}, {0, 0.8, 0.85}};
u = 0.6; v = 0.4;
rhit = quadrilateral[u, v, vertices];
rd = Normalize[{-3, 4, -12}]; (* a Pythagorean quadruple *)
ro = rhit - rd;

Clear[u, v, t];
sol = NSolve[Thread[quadrilateral[u, v, vertices] == ro + t rd], {u, v, t}, Reals];
sol = sol[[If[0 ≤ (t /. sol[[1]]) ≤ 1 && (t /. sol[[1]]) < (t /. sol[[2]]), 1, 2]]];
tx = t /. sol;

Print@SetPrecision[
    Join[sol, {rayorigin → ro, raydirection → rd, intersection → ro + tx * rd}], 10]

scale = 0.13; rscale = 0.002; uvsplits = 50;
labelsg = {Graphics3D[Table[{RGBColor[0.2, 0.6, 0.6],
    Specularity[Blue, 10], Sphere[n, 0.15 * scale]}, {n, vertices}]], Graphics3D/@
    MapIndexed[{White, Text[{"00", "10", "11", "01"}][[#2[[1]]]], #1, BaseStyle →
        {FontFamily → "Arial", FontSize → 14, TextAlignment → Center}]] &, vertices]]];

grst = ParametricPlot3D[quadrilateral[u, v, vertices], {u, 0, 1}, {v, 0, 1} (*,
    Mesh→30 *) , Mesh → 9, MeshStyle → {Directive[Thickness[0.0003], #] &/@ {Blue, Red}},
    PlotStyle → Directive[{Opacity[0.5], Cyan}], ColorFunction → "LightTerrain"];

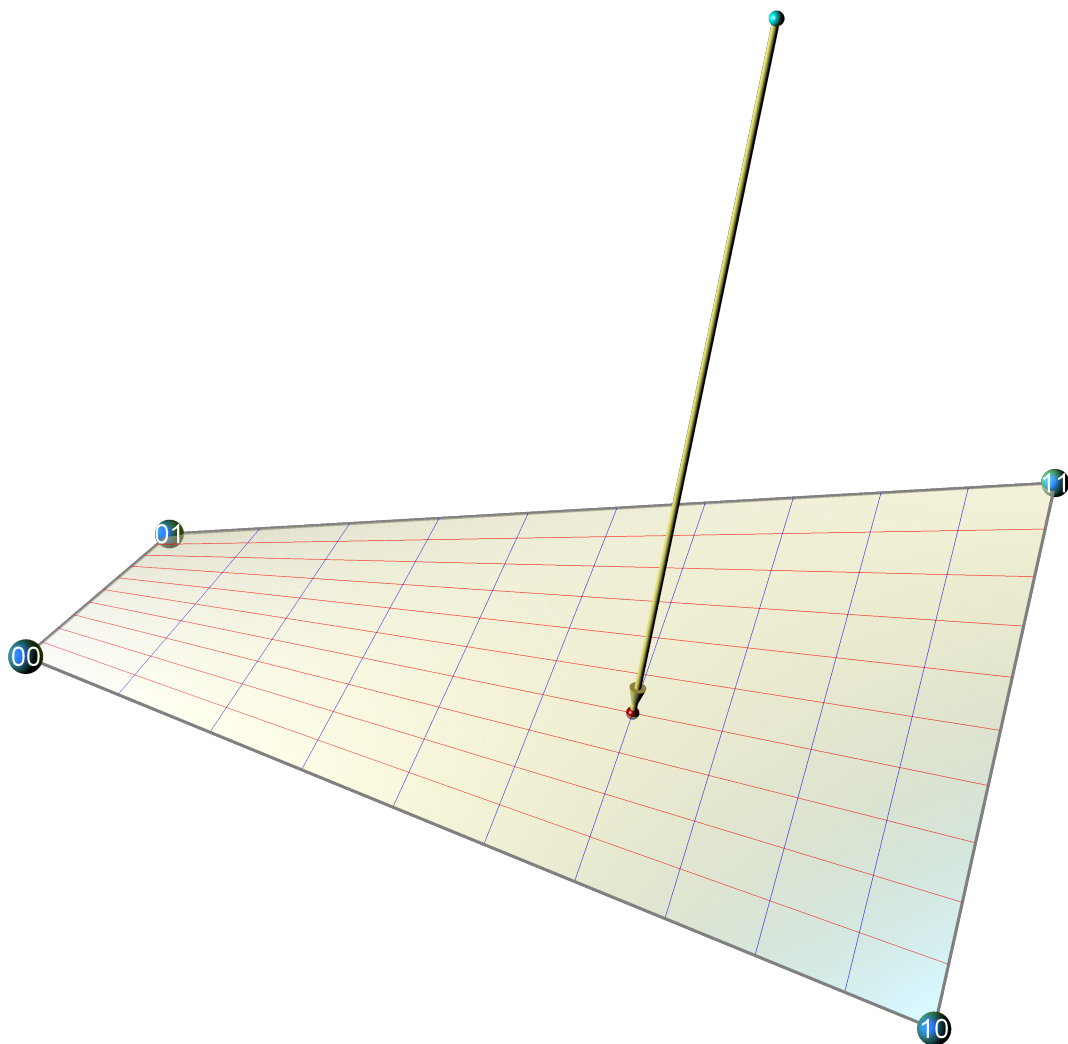
gvertices = Graphics3D[
    {Gray, Thickness[0.002], Opacity[1], Line[Append[vertices, First@vertices]]}];

grays = Graphics3D[{RGBColor[1., 1, 0.5], Arrowheads[10 * rscale],
    Arrow[Tube[{ro, ro + tx * rd}, Scaled[rscale]], 0 * rscale] ,
    Specularity[White, 50], Cyan, Sphere[ro, Scaled[2 * rscale]],
    Darker@Red, Sphere[ro + tx * rd, Scaled[2 * rscale]]}];

gn = Show[{labelsg, gvertices, grst, grays},
    ImageSize → 800, PlotRange → All, AxesStyle → Directive[Bold, 12],
    PlotStyle → Specularity[White, 500], AxesLabel → None, Axes → All, Boxed → False,
    Lighting → {"Point", Orange, {1, 1, 1}}, {"Point", White, {-4, -4, 4}}];
gn // Print

```

```
{u → 0.6000000000, v → 0.4000000000, t → 1.000000000,  
 rayorigin → {0.8787692308, 0.06030769231, 1.671076923},  
 raydirection → {-0.2307692308, 0.3076923077, -0.9230769231},  
 intersection → {0.6480000000, 0.3680000000, 0.7480000000}}
```



## (ray $\cap$ quad) in world coordinates

We find coefficients for the quadratic equation for **u** (section **8.4 Code**) as follows:

1. find a volume of the corresponding parallelepiped (section **8.2 GARP Details**) and set it to 0
2. It yields coefficients given by **polu**
3. We guesstimate the corresponding vector expressions for a, c, and a+b+c and
4. Verify that such expressions are equal to **polu** terms.

---

```

In[ ]:= Clear[p, o, d, u, v, t];
v4 = Table[Subscript[p, 10 * i + j], {i, 4}, {j, 3}]; (* 4 corner points *)
ro = Table[Subscript[o, i], {i, 3}]; (* ray origin *)
rd = Table[Subscript[d, i], {i, 3}]; (* ray direction *)
p12 = lerp[v4[[1]], v4[[2]], u];
p43 = lerp[v4[[4]], v4[[3]], u];
(* 1. *)
parallelepipedVolume[{l0_, ld_}, {m0_, md_}] := (m0 - l0).ld * md;
(* 2. *)
polu =
  FullSimplify[CoefficientList[parallelepipedVolume[{ro, rd}, {p12, p43 - p12}], u]];
polu // TraditionalForm // Print

(* 3. *)

(* a term *)
terma = Cross[v4[[1]] - v4[[4]], rd].(v4[[1]] - ro);
(* c term (it does not depend on ro) *)
termc = Cross[v4[[4]] - v4[[3]], v4[[1]] - v4[[2]].rd;
(* sum of a,b,c is *)
terms = Cross[v4[[2]] - ro, ro - v4[[3]].rd;
(* or *)
terms = Cross[v4[[2]] - ro, v4[[2]] - v4[[3]].rd;

(* 4. *)

{Expand[polu[[1]] == terma],
 Expand[polu[[3]] == termc], Expand[Total[polu] == terms]} // Print
{d3 (o2 (p11 - p41) + p12 (p41 - o1) + p42 (o1 - p11)) +
 d2 (p13 (o1 - p41) + o3 (p41 - p11) + p43 (p11 - o1)) + d1 (o3 (p12 - p42) + p13 (p42 - o2) + p43 (o2 - p12)),
 d3 (o2 (-p11 + p21 - p31 + p41) + o1 (p12 - p22 + p32 - p42) - p11 p32 + p12 (p31 - 2 p41) + p22 p41 + (2 p11 - p21) p42) +
 d1 (o3 (-p12 + p22 - p32 + p42) + o2 (p13 - p23 + p33 - p43) - p12 p33 + p13 (p32 - 2 p42) + p23 p42 + (2 p12 - p22) p43) +
 d2 (o3 (p11 - p21 + p31 - p41) + o1 (-p13 + p23 - p33 + p43) - p13 (p31 - 2 p41) - p23 p41 + p11 (p33 - 2 p43) + p21 p43),
 d3 ((p11 - p21) (p32 - p42) - (p12 - p22) (p31 - p41)) + d2 ((p13 - p23) (p31 - p41) - (p11 - p21) (p33 - p43)) +
 d1 ((p12 - p22) (p33 - p43) - (p13 - p23) (p32 - p42))}
{True, True, True}

```