Making Holes and Windows in Surfaces

Alan Horwitz

In this article, we demonstrate `makehole`, a program which removes points from any `Graphics` or `Graphics3D` picture whose coordinates satisfy a stated condition. We also demonstrate `transparent` and `makewindow`, programs which make an entire or a specific portion of an opaque surface into a transparent mesh. We use these programs to view the region of integration for a triple integral. This article uses *Mathematica* 5.2, but with minor modifications all three programs work in *Mathematica* 6, as well as earlier versions. The `makehole` program duplicates some of the functionality of the `RegionPlot3D` command and `RegionFunction` option of the `ParametricPlot3D` command in Version 6, while the `transparent` program behaves like the `PlotStyle → None` option of the `Plot3D` command (see Editor's Note for a demonstration of the *Mathematica* 6 code).

■ Introduction

Given a 2D or 3D object, it is useful to view its cutaway image along an intersecting curve or surface. For example, we could cut an object along planes where x, y, or z are constant by adjusting the `PlotRange` parameters. For cutting along a general curve or outside a surface, we introduce `makehole`, a program that removes vertices in `Polygon`, `Line`, and `Point` graphics primitives whose coordinates satisfy a stated condition (e.g., an equation, inequality, or compound inequality). We also introduce `transparent` and `makewindow`, which make an opaque surface into a transparent mesh by replacing each `Polygon` primitive with line segments around the boundary of the polygon. Viewing a region of integration for a triple integral in rectangular coordinates is one application of these programs.

■ Making Surfaces Transparent

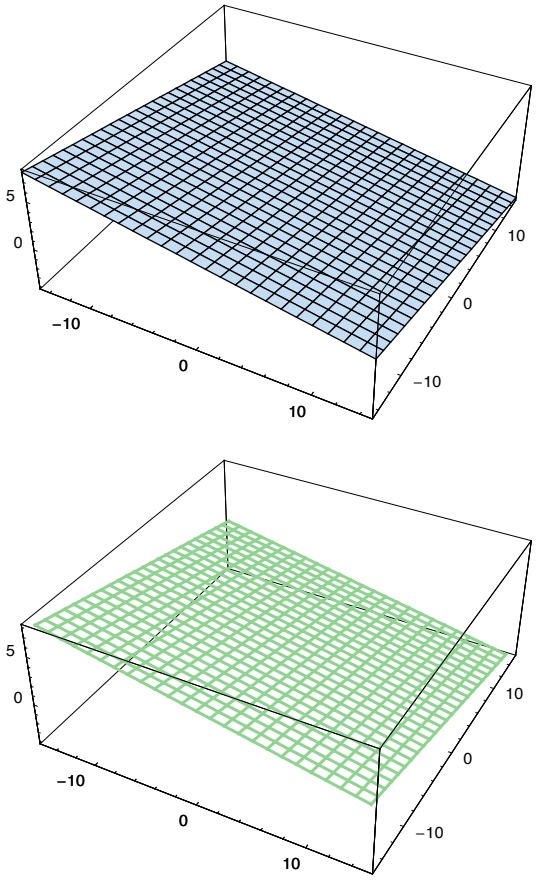
We first introduce `transparent`. To outline the boundary of a polygon, it repeats the first vertex at the end of each list of vertices and replaces all `Polygon` heads with `Line`. Unlike `WireFrame` in the *Shapes* package, `transparent` has adjustable `PlotStyle` options and works on graphics with either `Graphics` or `Graphics3D` heads.

`transparent[object_,`*`options___`*`]`	*object* must have either a `Graphics` or `Graphics3D` head; options are the same as `PlotStyle` options

```
In[1]:= transparent[object_, options___] := Module[{plotstyle},
          plotstyle =
        PlotStyle /. {options} /. PlotStyle → Thickness[.005];
          object /. Polygon[x___] :>
         Flatten[{plotstyle, Line[Append[x, First[x]]]}]]
```

We use `Plot3D` to draw the $x + y + 5z = 10$ plane and `//Graphics3D` to convert the `SurfaceGraphics` head to `Graphics3D`. Then we apply `transparent` and show the result.

```
In[2]:= plane =
          Plot3D[1 / 5 (10 - x - y), {x, -15, 15}, {y, -15, 15}] // Graphics3D;
        newplane = transparent[plane,
           PlotStyle → {RGBColor[.5, .8, .5], Thickness[.006]}];
        Show[newplane]
```



■ Making Holes

Before introducing `makehole`, we demonstrate its main ideas by removing portions of the following `Graphics3D` object that lie on or above the $x + y + 5\,z = 10$ plane. The object consists of two polygons.

```
In[6]:= object =
          Graphics3D[{Polygon[{{2, 0, 0}, {0, 2, 0}, {0, 0, 2}, {0, 0, 0}}],
            Polygon[{{5, 5, 0}, {0, 5, 1}, {5, 0, 1}}]},
           {Axes → True, AxesLabel → TraditionalForm /@ {x, y, z}}];
```

The condition $x + y + 5\,z \geq 10$ is expressed as `g[u,v,w]` in the following. Also, `{u_,v__}/; g[u,v]` is used to represent the vertices with either two or three coordinates that satisfy condition `g`. `DeleteCases` removes vertices from the polygons that satisfy the condition, as well as polygons with no remaining vertices.

```
In[7]:= condition = x + y + 5 z ≥ 10;
        g[u_, v_, w_] := condition /. x → u /. y → v /. z → w;
        newobject = object /. {poly : Polygon[_] :>
             DeleteCases[poly, {u_, v__} /; g[u, v], {2}]};
        output = DeleteCases[newobject, Polygon[{}], ∞];
```

We view the output in `InputForm` and observe that `{0,0,2}` and all vertices of the second polygon were removed from the original object because they satisfied $x + y + 5\,z \geq 10$.

```
In[11]:= InputForm[output]

Out[11]//InputForm=
Graphics3D[{Polygon[{{2, 0, 0}, {0, 2, 0}, {0, 0, 0}}]},
 {Axes -> True, AxesLabel -> {"x", "y", "z"}}]
```

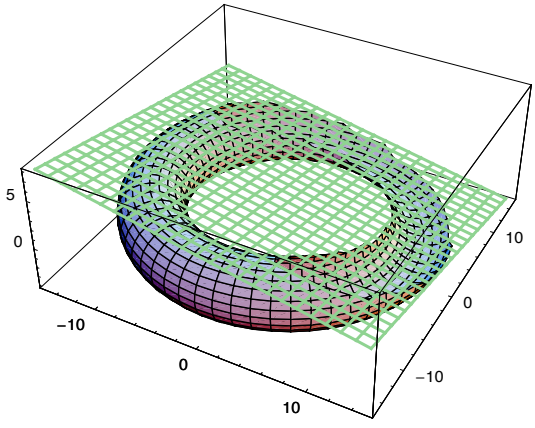
We note that `Polygon` primitives with fewer than three remaining vertices will not be displayed by `Show`.

To demonstrate that this strategy works for more complicated objects, we generate a torus (suppressing the plot using `DisplayFunction → Identity`), then use the same commands to remove the portions on and above $x + y + 5\,z = 10$.

```
In[12]:= torus = ParametricPlot3D[
            {(12 + 3 Cos[t]) Cos[s], (12 + 3 Cos[t]) Sin[s], 3 Sin[t]},
             {s, 0, 2 Pi}, {t, 0, 2 Pi}, PlotPoints → {60, 25},
            DisplayFunction → Identity];
         condition = x + y + 5 z ≥ 10;
         g[u_, v_, w_] := condition /. x → u /. y → v /. z → w;
         newobject = torus /. {poly : Polygon[_] :>
              DeleteCases[poly, {u_, v__} /; g[u, v], {2}]};
         output = DeleteCases[newobject, Polygon[{}], ∞];
```

We show the results with the transparent $x + y + 5z = 10$ plane.

```
In[17]:= Show[output, newplane, DisplayFunction → $DisplayFunction]
```



The complete `makehole` program, which follows, removes vertices with two and three coordinates that satisfy a stated condition from within `Polygon`, `Line`, and `Point` primitives.

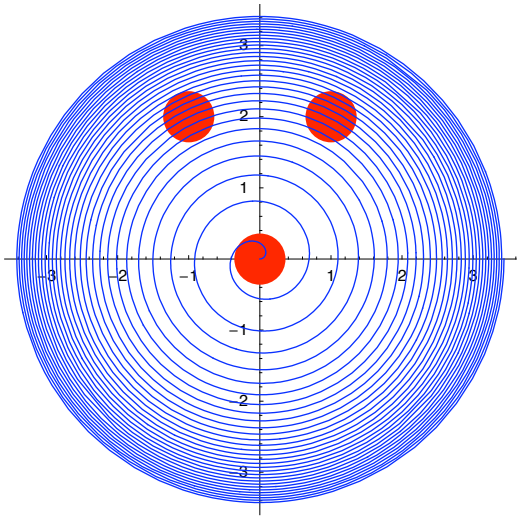
`makehole[`*object_*`, {`*x_*`, `*y_*`, `*condition__*`}]`	removes all points (x, y) from the 2 D *object* that satisfy *condition*; object must have a `Graphics` head
`makehole[`*object_*`, {`*x_*`, `*y_*`, `*z_*`, `*condition__*`}]`	removes all points (x, y, z) from the 3 D *object* that satisfy *condition*; *object* must have a `Graphics3D` head

```
In[18]:= makehole[object_, {x_, y_, condition_}] :=
         Module[{g},
         g[u_, v_] := condition /. x → u /. y → v;
         makehole[object, g]];

         makehole[object_, {x_, y_, z_, condition_}] :=
         Module[{g},
         g[u_, v_, w_] := condition /. x → u /. y → v /. z → w;
         makehole[object, g]];

         makehole[object_, g_] :=
         Module[{shape, partitwo, templabel},
         (* This part reformats the Line primitives so that
                 adjacent points are in their own Line primitive. *)
```

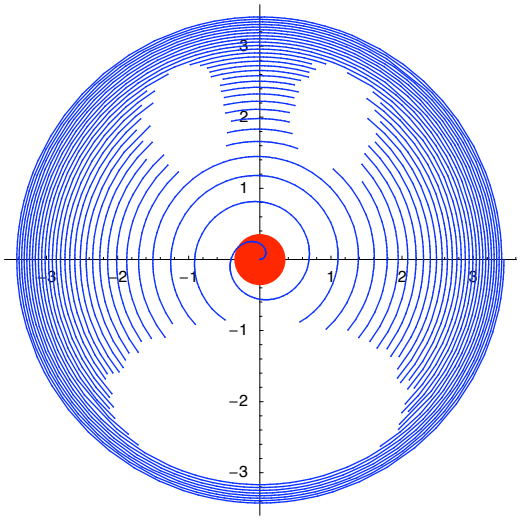
```
shape = object /. Line[t_] → templabel[t];
partitwo[shape_] := Map[Line, Partition[shape, 2, 1]];
  shape = shape /.
     ({templabel : templabel[_] :> Map[partitwo[#] &, templabel]} //
       Flatten) /. templabel[t_] → t;
    shape = shape /. {line : Line[_] :> DeleteCases[
       line, {u_, v__} /; g[u, v], {2}],
     poly : Polygon[_] :> DeleteCases[poly,
       {u_, v__} /; g[u, v], {2}],
     point : Point[_] :> DeleteCases[point,
       {u_, v__} /; g[u, v], {1}]};
    shape = Fold[DeleteCases[#1, #2, Infinity] &, shape,
                 {Line[{}], Polygon[{}], Point[]}] ]
```

To demonstrate `makehole` on graphics with `Graphics` heads, we use it to remove regions inside three ellipses from the graph of a spiral and three points. In order to remove a union of regions, we use `||` to represent "or," so that points that satisfy any of the three inequalities are removed.

```
In[21]:= graph =
   ParametricPlot[Log[1 + t] {Cos[2 Pi t], Sin[2 Pi t]}, {t, 0, 30},
    PlotPoints → 200, PlotStyle → RGBColor[0, 0, 1],
    DisplayFunction → Identity];
points = Graphics[{PointSize[.1], RGBColor[1, 0, 0], Point[{1, 2}],
    Point[{-1, 2}], Point[{0, 0}]}, {Axes → True}];
picture = Show[{points, graph}, PlotRange → All,
  AspectRatio → Automatic]
```



```
In[24]:= condition = (2 (x - 1) ^2 + (y - 2) ^2 < .5 ) ||
           (2 (x + 1) ^2 + (y - 2) ^2 < .5) || (x^2 + 3 (y + 2) ^2 < 4 );
         Show[makehole[picture, {x, y, condition}],
          Axes → True, AxesOrigin → {0, 0}]
```



■ Making Windows in Surfaces

The `makewindow` program replaces `Polygon` with `Line` primitives if one of the vertices satisfies a stated condition.

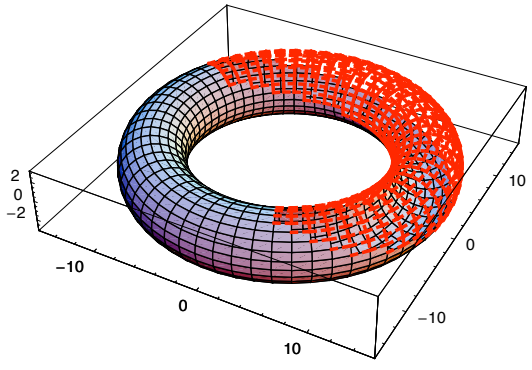
`makewindow`[*object_*, {*x_*, *y_*, *condition__*}, *options___*]	*object* must have a `Graphics` head
`makewindow`[*object_*, {*x_*, *y_*, *z_*, *condition__*}, *options___*]	*object* must have a `Graphics3D` head

options are the same as `PlotStyle` options, including RGBColor, `Thickness`, and `Dashing`

```
In[26]:= makewindow[object_, {x_, y_, condition_}, options___] :=
         Module[{g},
         g[u_, v_] := condition /. x → u /. y → v;
         makewindow[object, g, options]];

         makewindow[object_, {x_, y_, z_, condition_}, options___] :=
         Module[{g},
         g[u_, v_, w_] := condition /. x → u /. y → v /. z → w;
         makewindow[object, g, options]];

         makewindow[object_, g_, options___] :=
         Module[{shape, window, plotstyle},
             plotstyle = PlotStyle /. {options} /.
             (PlotStyle → {Thickness[.005], RGBColor[.5, .5, .5]});
                 window[Polygon[{x___, {u_, v__}, y___}] /; g[u, v]] :=
            Flatten[{plotstyle, Line[{x, {u, v}, y}]}];
                 window[Polygon[{x___, {u_, v__}, y___}] /; ! g[u, v]] :=
            Polygon[{x, {u, v}, y}];
         shape = object /. {poly : Polygon[_] :> window[poly]}]
```

To show how it works, we use the torus and the condition $x + y + 5\,z \geq 10$ to create a mesh window on portions lying above the corresponding plane. We use `PlotStyle` options to control the color and thickness of the mesh and to introduce dashing.

```
In[29]:= picture = ParametricPlot3D[
            {(12 + 3 Cos[t]) Cos[s], (12 + 3 Cos[t]) Sin[s], 3 Sin[t]},
             {s, 0, 2 Pi}, {t, 0, 2 Pi}, PlotPoints → {60, 25},
            DisplayFunction → Identity];
         newpicture = makewindow[picture,
            {x, y, z, x + y + 5 z ≥ 10}, PlotStyle →
             {RGBColor[1, 0, 0], Thickness[.007], Dashing[{.01, .015}]}];
         Show[newpicture, DisplayFunction → $DisplayFunction]
```



■ Showing the Region of Integration for a Triple Integral

We use `makehole` and `makewindow` to draw a region corresponding to the triple integral, $\int_{-2}^{2}\int_{0}^{4-y^2}\int_{0}^{x} 1 \; dz \; dx \; dy$. To visualize the region, we draw many of the bounding surfaces and then impose the three conditions $0 \le z \le x$, $0 \le x \le 4 - y^2$, and $-2 \le y \le 2$ from the limits of integration.

We first draw bounding surfaces for the region of integration. The surfaces drawn with `Plot3D` are converted to a `Graphics3D` head using `//Graphics3D`. The blue bounding surface $z = 0$ and the purple surface $z = x$ are suppressed.

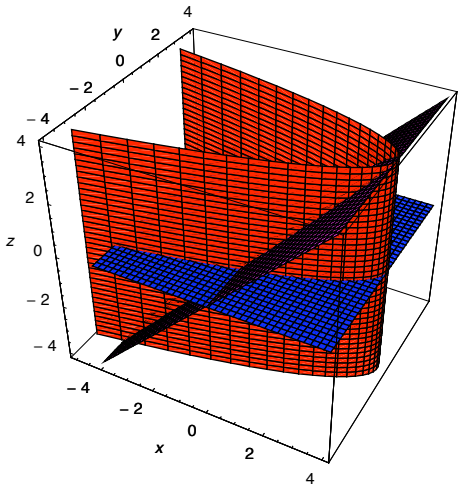
```
In[32]:= zeqzero =
           Plot3D[{0., RGBColor[0, 0, 1]}, {x, -4, 4},
           {y, -4, 4}, AxesLabel → TraditionalForm /@ {x, y, z},
               PlotPoints → 40, DisplayFunction → Identity] // Graphics3D;
         zeqx = Plot3D[{x, RGBColor[1, 0, 1]}, {x, -4, 4},
           {y, -4, 4}, AxesLabel → TraditionalForm /@ {x, y, z},
               PlotPoints → 40,
               DisplayFunction → Identity] // Graphics3D;
```

To sketch the surface $x = 4 - y^2$, we view it as a parametrized surface between cross-sectional curves on planes $z = -4$ and $z = 4$. We color the surface red and suppress the plot.

```
In[34]:= f[t_] := {4 - t^2, t, -4};
         g[t_] := {4 - t^2, t, 4};
         h[{s_, t_}] := f[t] (1 - s) + g[t] s;
         redsurface := Append[h[{s, t}], RGBColor[1, 0, 0]];
         xeq4minusysqr =
           ParametricPlot3D[redsurface // Evaluate, {s, 0, 1}, {t, -3, 3},
                PlotPoints → 40, Lighting → False,
            DisplayFunction → Identity];
```

The region of integration is enclosed by the surfaces. We omit the surfaces $x = 0$, $y = -2$, and $y = 2$ from the image because the present region does not cross these planes.

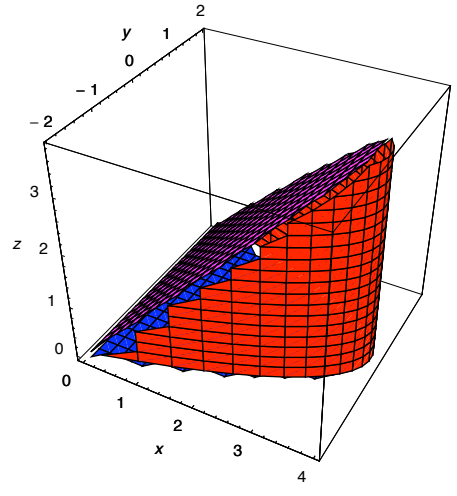
```
In[39]:= picture = Show[xeq4minusysqr, zeqzero,
           zeqx, DisplayFunction → $DisplayFunction,
           AxesLabel → TraditionalForm /@ {x, y, z}]
```



To isolate the region of integration, we use `makehole` to remove all polygons that lie outside the region.

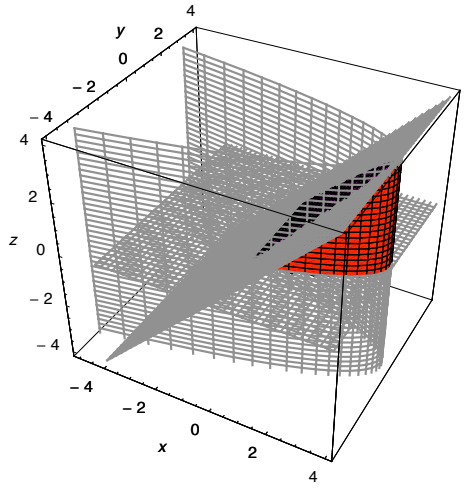
The condition `!intgrlmts` removes all points (x, y, z) whose coordinates do not satisfy each of $0 \le z \le x$, $0 \le x \le 4 - y^2$, and $-2 \le y \le 2$.

```
In[40]:= intgrlmts = ((0 <= z <= x) && (0 <= x <= 4 - y^2) && (-2 <= y <= 2));
         region = makehole[picture, {x, y, z, ! intgrlmts}];
         Show[region]
```



To display the region inside the original image, we use `makewindow` to make everything outside the region transparent.

```
In[43]:= intgrlmts = ((0 ≤ z ≤ x) && (-2 ≤ y ≤ 2) && (0 ≤ x ≤ 4 - y^2));
         region2 = makewindow[picture, {x, y, z, ! intgrlmts}];
         Show[region2, BoxRatios → Automatic, PlotRange → All]
```



■ Conclusion

We have demonstrated programs for making holes and windows in portions of two- and three-dimensional objects which satisfy one or more specified inequalities. Applications include slicing an opaque surface along an intersecting surface and viewing the region of integration for a triple integral.

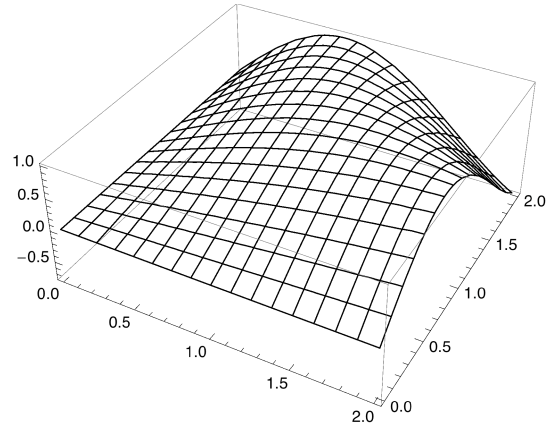
■ Editor's Note

This article describes techniques for working with legacy graphics in *Mathematica*. The code would need to be modified to work with `GraphicsComplex` and other extensions to the *Mathematica* graphics language in Version 6. *Mathematica* 6 also contains new features that overlap with the functionality described in this article.

Use `PlotStyle → None` to generate wireframe surfaces.

```
In[1]:= Plot3D[Sin[x y], {x, 0, 2}, {y, 0, 2}, PlotStyle → None]
```

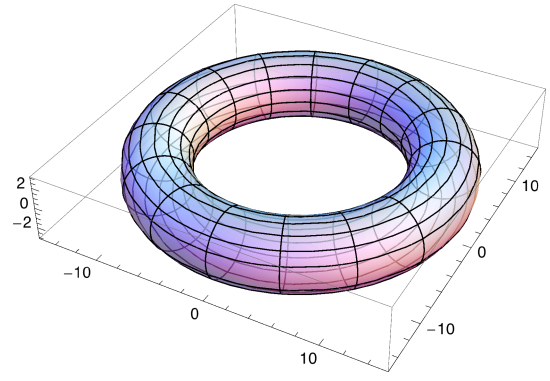
Out[1]=



Use `Opacity` to create partially transparent surfaces.

```
In[2]:= ParametricPlot3D[
        {(12 + 3 Cos[t]) Cos[s], (12 + 3 Cos[t]) Sin[s], 3 Sin[t]},
        {s, 0, 2 π}, {t, 0, 2 π}, PlotStyle → Opacity[0.7]]
```

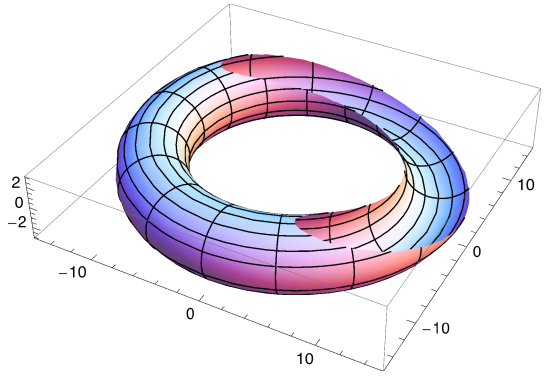
Out[2]=



Use `RegionFunction` to remove portions of a surface.

```
In[3]:= ParametricPlot3D[{(12 + 3 Cos[t]) Cos[s],
          (12 + 3 Cos[t]) Sin[s], 3 Sin[t]}, {s, 0, 2 π}, {t, 0, 2 π},
        RegionFunction → Function[{x, y, z, s, t}, x + y + 5 z ≤ 10]]
```

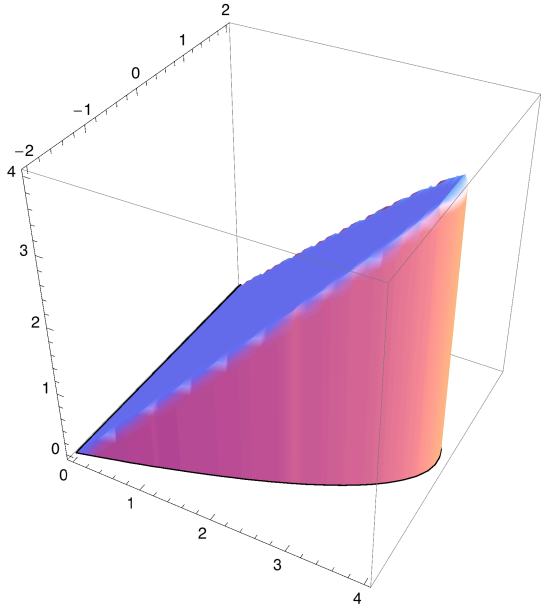
Out[3]=



Use `RegionPlot3D` to show a region defined by inequalities.

```
In[4]:= RegionPlot3D[0 ≤ z ≤ x && 0 ≤ x ≤ 4 - y^2 && -2 ≤ y ≤ 2, {x, 0, 4},
          {y, -2, 2}, {z, 0, 4}, Mesh → None, PlotPoints → 30]
```

Out[4]=



■ References

A. Horwitz, "Making Holes and Windows in Surfaces," *The Mathematica Journal*, 2011. dx.doi.org/doi:10.3888/tmj.10.4–4.

About the Author

Alan Horwitz received a B.S. from Ohio State University in 1980 and an M.A. in 1985 and a Ph.D in 1988 from SUNY at Stonybrook. He is currently a faculty member in the Mathematics Department at Marshall University.

Alan Horwitz
Dept. of Math, Marshall University
Huntington, West Virginia 25755
horwitz@marshall.edu