

TURBOPK

USER MANUAL

March 2014

Patrick Buckley

SURVICE Engineering LLC

Contents

1.0	Installation	3
2.0	First Example Calculation	4
3.0	Rayleigh Burst Point Option	14
3.1	Rayleigh Distribution In Weapon Coordinates	19
4.0	HEI Projectile Grid	28
5.0	Projectile Parallel Shotline Options	31
6.0	Parallel Shotlines - Add Armor Option	37
7.0	Single Burst Point Option	41
8.0	Graphics Display Control	44
9.0	Edit Menu	45
9.0	Warhead Files	46
10.0	Shotline Inspector	49
11.0	Burst Point Grid Polar Coordinates	58
12.0	Rectangle of Trajectories Option	60
13.0	Air Blast Model	71
14.0	Miscellaneous	79

1.0 Installation

TurboPK comes as a single "ZIP" file with a name like "TurboPK_v1.1.4_64bit.zip." The "v1.1.4" in the same indicates the current version number (yours may be different), and "64bit" indicates the computer word size. Other names may have "32bit" or "Linux", for which 64-bit is the only version. There are a number of ZIP utilities that can unzip this file. If you don't have one, we recommend IZARC (www.izarc.org/download.html) because that utility was used to create the ZIP file in the first place.

Unzip the TurboPK ZIP to the folder of your choice, for example c:\TurboPK_Standalone. (Do *not* unzip it to C:\Windows or any of its sub-folders.)

Unzipping the file will create the following contents in the folder you selected (Figure 1):






 Vehide	4/1/2013 10:24 AM	File folder	
 QtCore4	3/8/2013 6:51 PM	DLL File	2,230 KB
 QtGui4	11/2/2010 8:14 PM	DLL File	7,976 KB
 QtOpenGL4	11/2/2010 8:21 PM	DLL File	661 KB
 turbopk	4/1/2013 10:26 AM	Application	695 KB

Figure 1 - TurboPK ZIP contents.

The base folder has the application file (turbopk.exe) and three DLLs (QtCore4, QtGui4, and QtOpenGL4). The DLLs are from the Qt software development system, which was used to create TurboPK. Unfortunately there are many different versions of the Qt DLLs, not all of which are compatible with TurboPK. And the ones that are compatible with TurboPK are not necessarily compatible with other Qt-based applications you may have on your machine. So we *strongly* advise that you keep the TurboPK executable and its DLLs in the same folder, and that folder *not* be C:\Windows or any of its sub-folders. If you want to move TurboPK to some other folder then make sure the Qt DLLs are also moved to the new folder.

Note that your particular ZIP may contain other files, including other DLLs which should be treated in the same way.

To create a TurboPK icon on your desktop just drag-and-drop the executable file name onto your desktop.

2.0 First Example Calculation

Launch TurboPK by double-clicking on the TurboPK.exe file name, or by double-clicking on its desktop icon. You should see the following startup window:

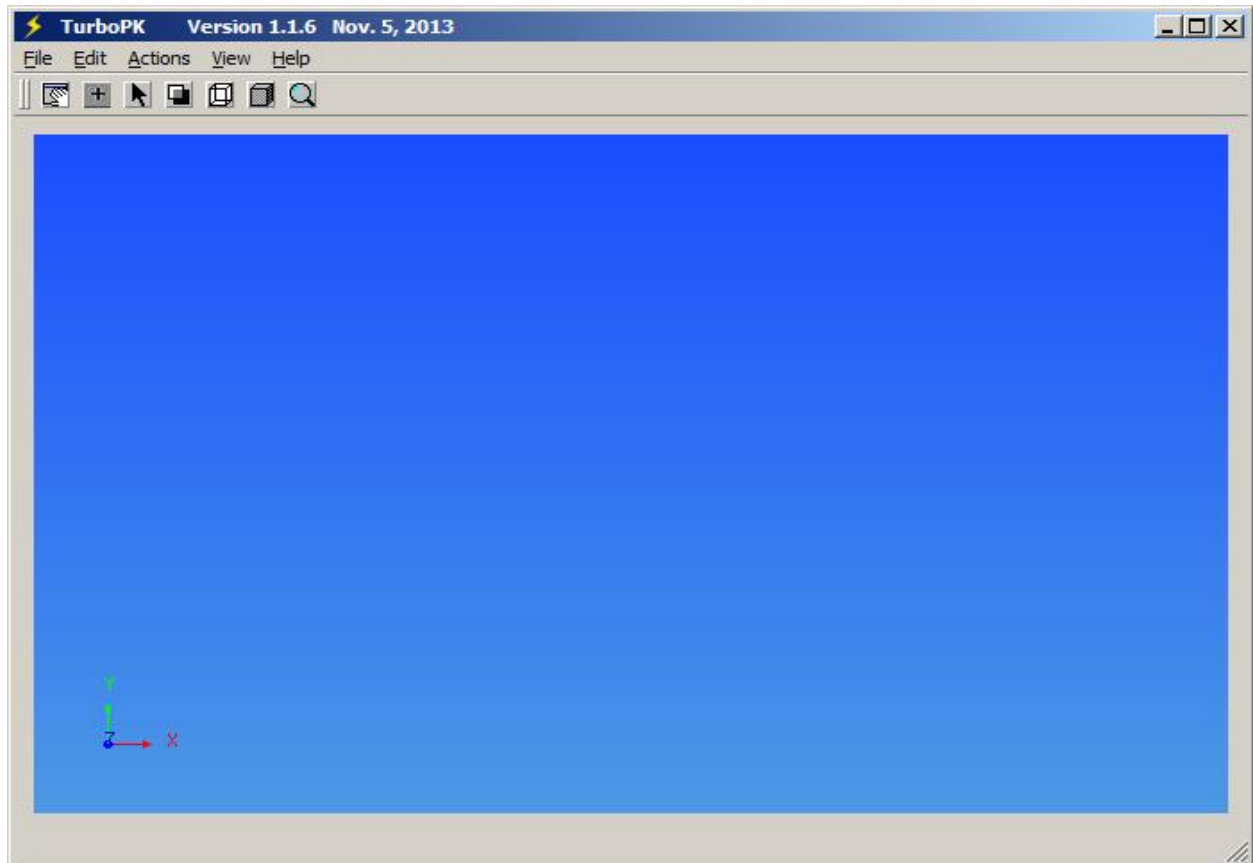


Figure 2 - TurboPK startup window.

From the **File** menu select the **File Open File Set** item (Figure 3).

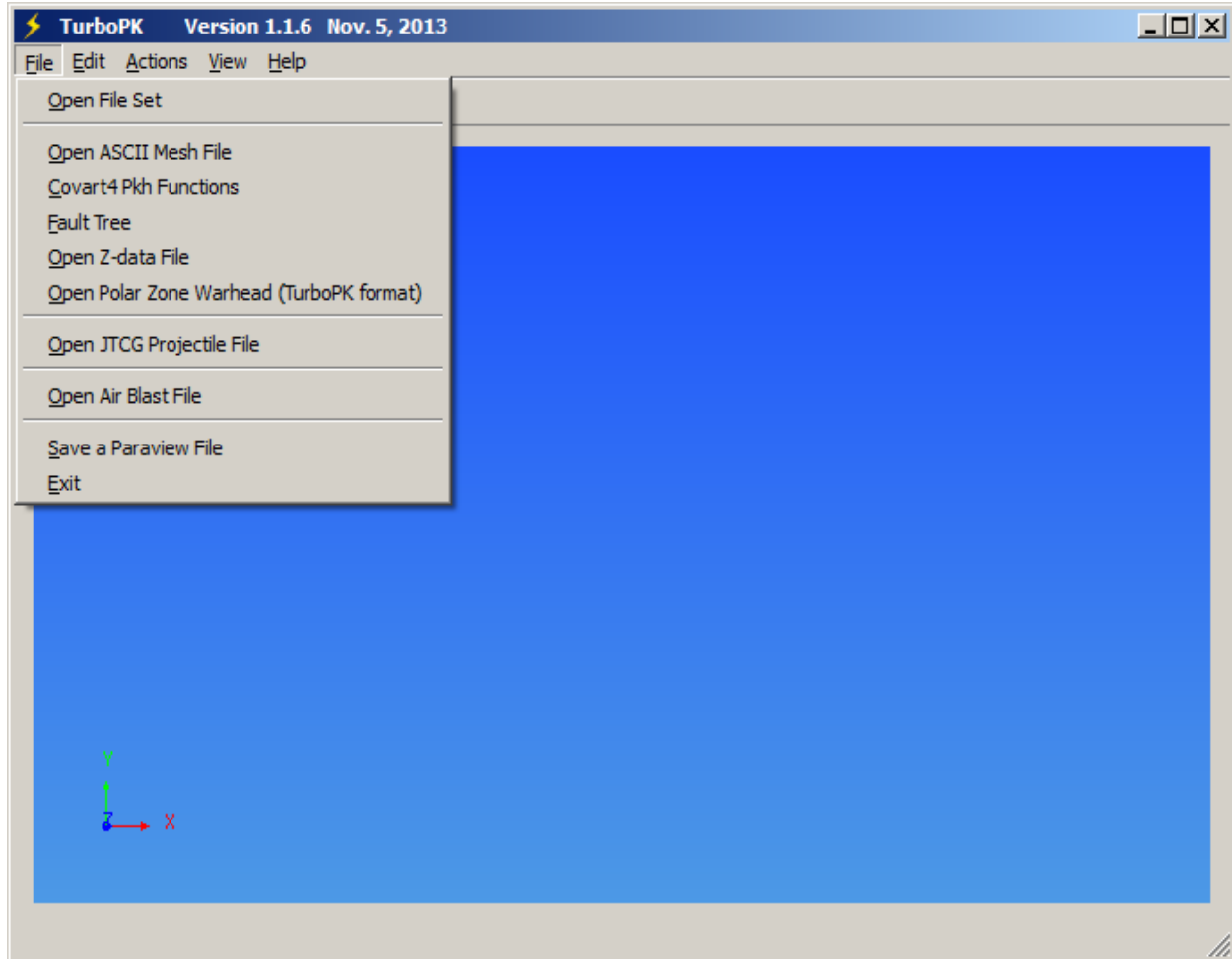


Figure 3 - Opening a set of files.

This pops up a *folder* selection box (Figure 4). In the case of Figure 4 the folder **Vehicle** is selected. **Vehicle** is a sub-folder provided in the distribution zip file. This will cause TurboPK to load several files from the **Vehicle** sub-folder.

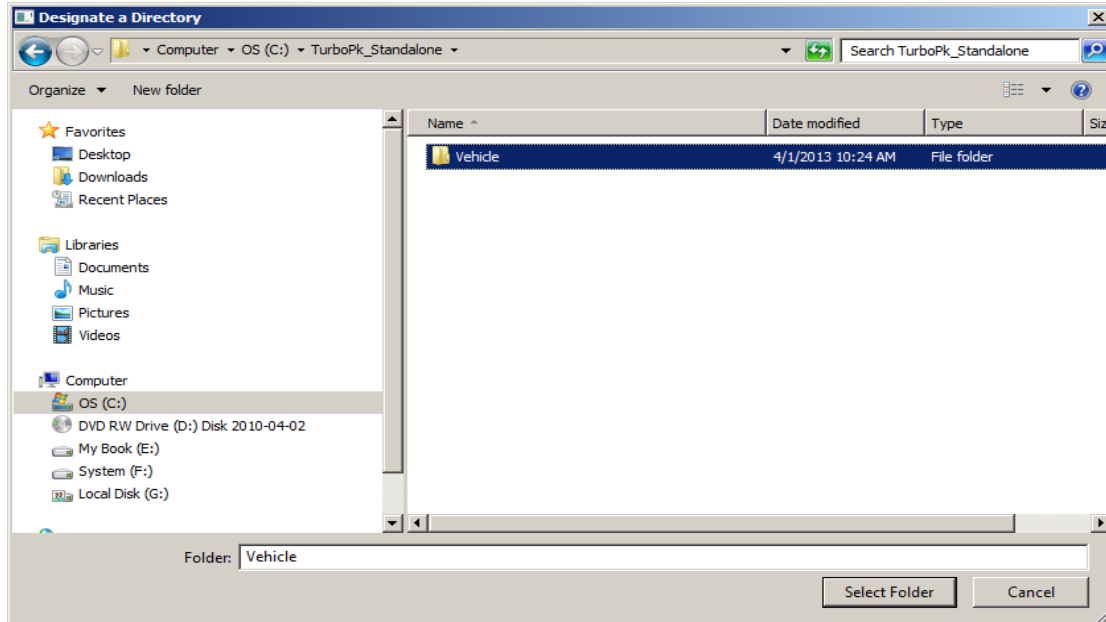


Figure 4 - Selecting the Vehicle sub-folder.

The files loaded into TurboPK include "vehicle.msh," which is the target geometric model. "Vehicle.pkh" is the set of damage functions for the vulnerable components in the target model. "Vehicle.mv" contains the fault tree describing the logical interconnections between vulnerable components. "ExampleWhd_ZDATA.zdata" is a ZDATA description of a fragmentation warhead. "Example.prjc" is a JTCG-projectile file, and "blast" is an air blast vulnerability file.

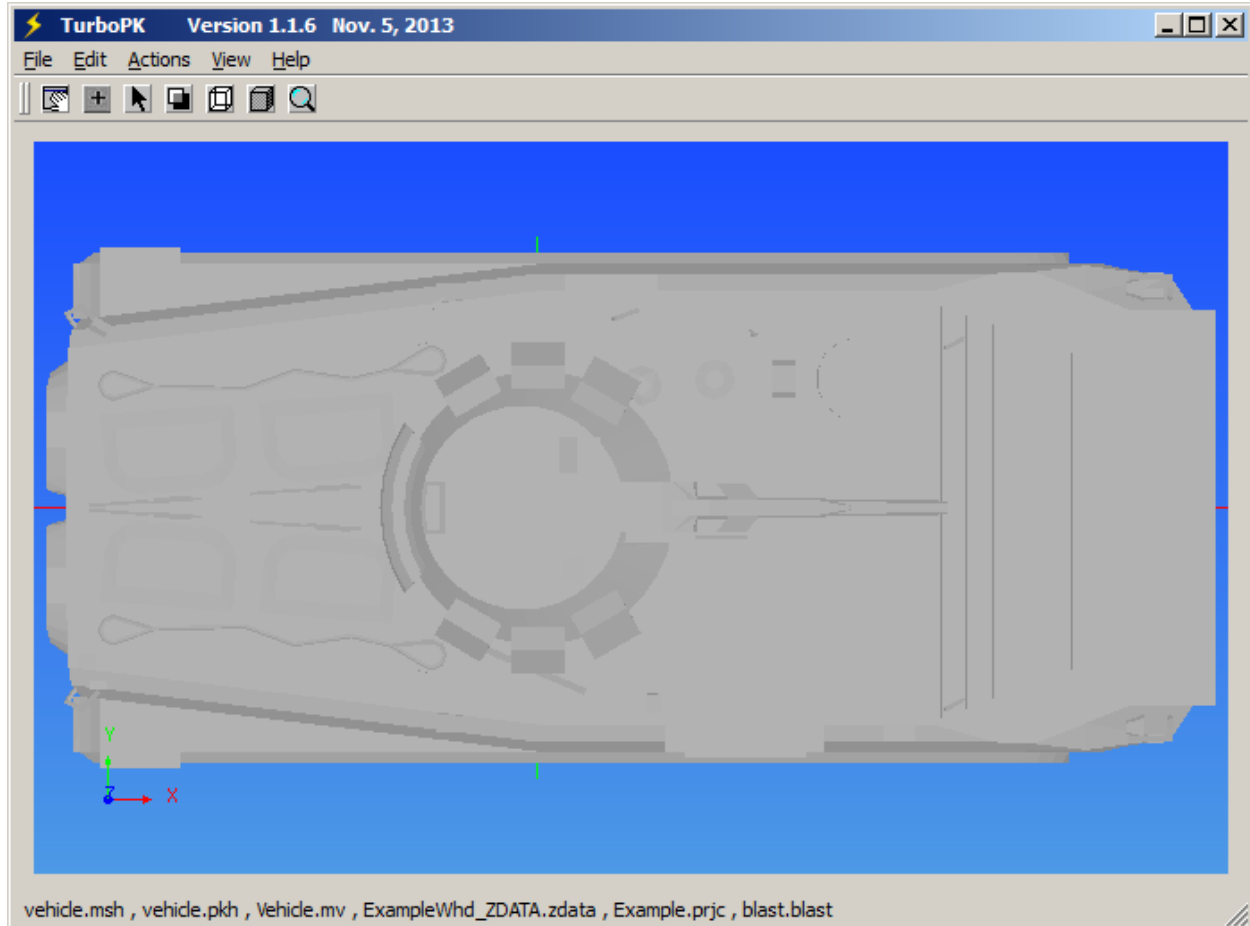


Figure 5 - Files loaded from the "Vehicle" folder.

ZDATA files contain information about fragment masses, but not fragment shapes or fragment materials, so the dialog box shown in Figure 6 pops up asking the user to define a fragment shape and to pick a fragment material. (See Section 9.0 for more details.) The dialog box titled "Fragment Parameters" presents five shape options and a drop-down list for specifying fragment material. The edit controls that are grayed out become active when their associated fragment shape is selected.

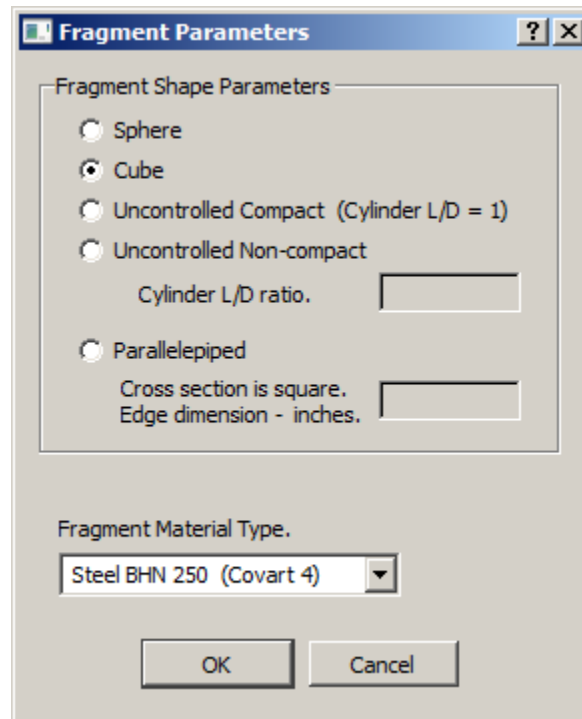


Figure 6 - Fragment shape dialog.

Each time a warhead file is loaded the user is prompted to enter a "TNT Equivalent Weight" for air blast calculations (Figure 7). A description of the air blast algorithm is provided later. The "TNT Equivalent Weight" parameter in this case is the equivalent weight for pressure.

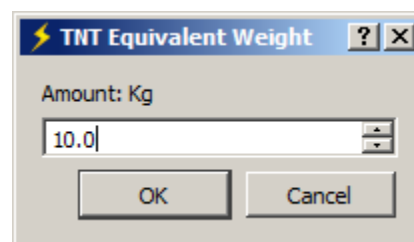


Figure 7 - Explosive weight dialog

The **File Open File Set** option is just a shortcut method for opening the files individually. The underlying code will go to the folder selected by the user and will load the *first* file of each type (.msh, .pkh, .mv, .pzw, and .prjc) it encounters.

The analysis options offered by TurboPK are found in the **Actions** menu. For this example the item **Actions...Burst Point Field** is selected (Figure 8). This pops up the dialog box labeled "Burst Point Field" (Figure 9). This option generates a set of burst points at a fixed height above the $z = 0.0$ plane in target coordinates.

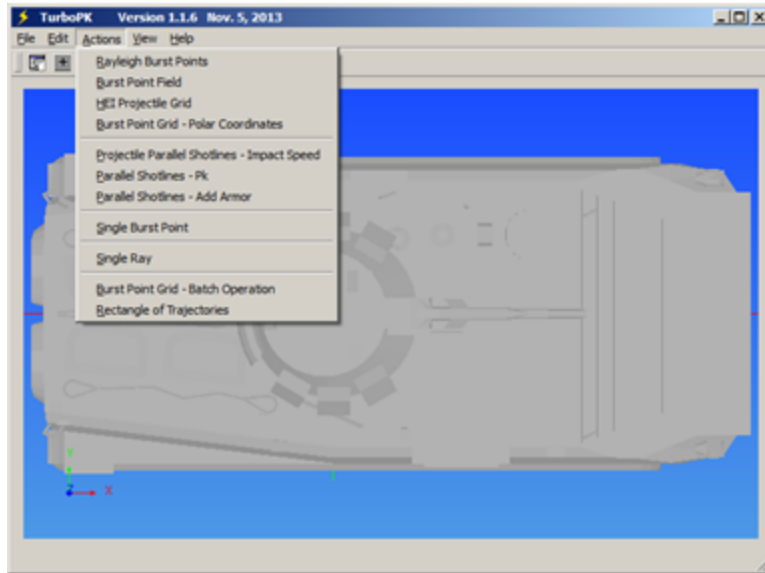


Figure 8 - Actions menu

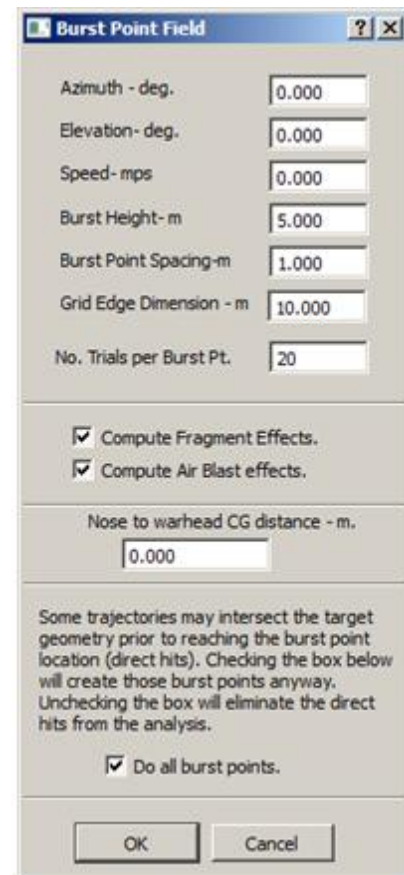


Figure 9 - Burstpoint field dialog

Azimuth and Elevation define the approach angles of the weapon with respect to the target at rest. Azimuth of 0.0 is "head-on." Elevation of 0.0 is parallel to the ground plane. Elevation of 90.0 is diving down onto the target. "Speed" refers to the weapon's speed. Burst height is the height of the burst point plane above the $z = 0.0$ plane in target coordinates. Burst point spacing is the distance between burst points as measured in the burst point plane. Grid Edge Dimension is the edge dimension of the burst point plane square. No. Trials per Burst Point is the number of point-burst Monte Carlo trials that are conducted at each burst location. "Nose to warhead CG distance" is the distance from the nose of the weapon in question to the center of gravity of the warhead. This is a *distance* so it is always positive. The check box "Do all burst points" controls whether or not certain burst point locations are generated. In particular, it is possible for a burst point to be located at a point beyond where its associated weapon trajectory would have already intersected the target geometry. Checking the box will cause these burst points to be generated in the grid even though the weapon would have hit the target first. Clicking the

Ok button in the dialog box causes TurboPK to generate the grid of burst points and do the burst point simulations. Results are computed as average P_K values per burst point and are presented as color-coded markers at the burst point locations (Figure 10).

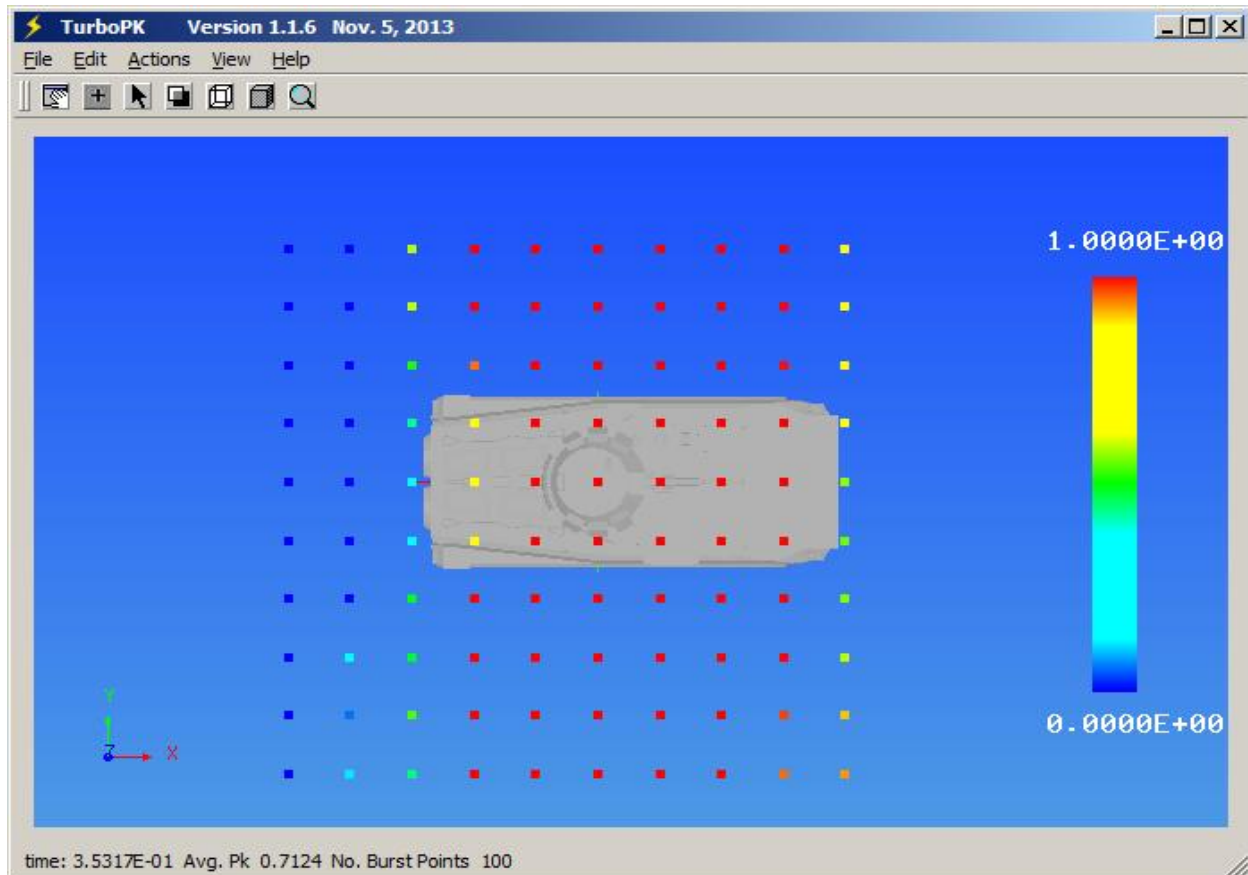


Figure 10 - Burst point field P_K values

A few summary statistics are presented in the bottom frame of the TurboPK main window. In this example there were 100 burst points. P_K averaged over the 100 burst points was 0.71. Run time was 0.35 seconds. The warhead in this example has a total of 2000 fragments, so each Monte Carlo sample point-burst involves simulating 2000 randomized fragment rays. It also involves a set of 14,400 "blast rays" generated at 3-degree intervals in polar angle and roll angle. Performing 20 Monte Carlo samples at each of 100 burst points therefore involves simulating a total of 3×10^7 rays.

A higher resolution version of the example calculation is shown in Figure 11. In this case the spacing between burst points was set to 0.1 meters, which results in 9800 burst points. This calculation involves roughly 3×10^9 fragment rays and required 22.5 seconds of calculation time.

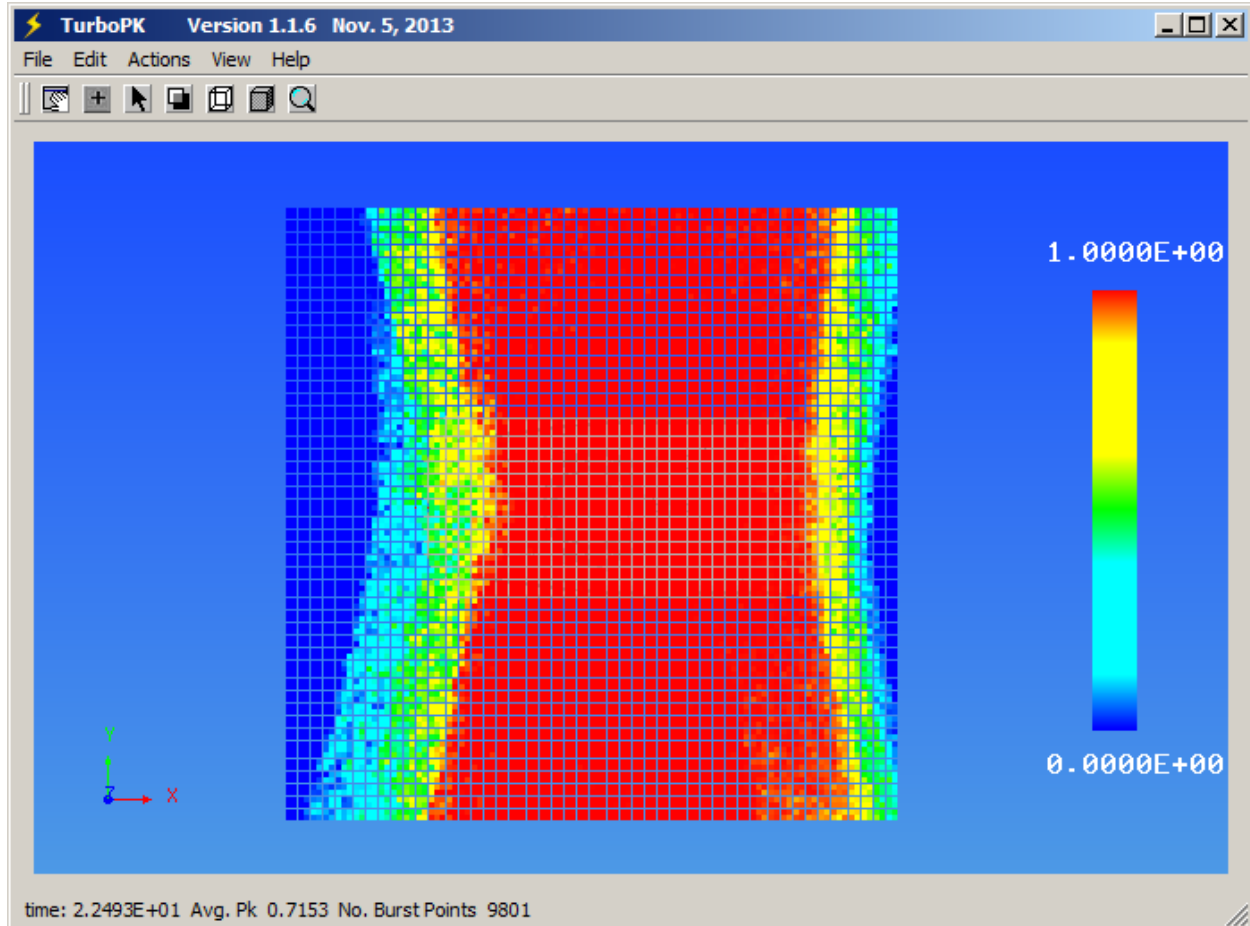


Figure 11 - Burst point spacing reduced to 0.1 meters.

Figure 12 shows the Windows Task Manager during the 9800 burst point run. It shows that TurboPK is using 100% of the CPU compute power. Core i7 processors have four full computing cores, but each core is "hyperthreaded" so two separate computational threads share each core's computing resources. Hyperthreading appears to Windows as eight separate computations running simultaneously, so its performance window reports the usage of eight "CPUs." TurboPK was built from the ground up to take advantage of multi-core processor designs through parallel programming techniques and Figure 12 indicates it is successful in using all cores at the same time.

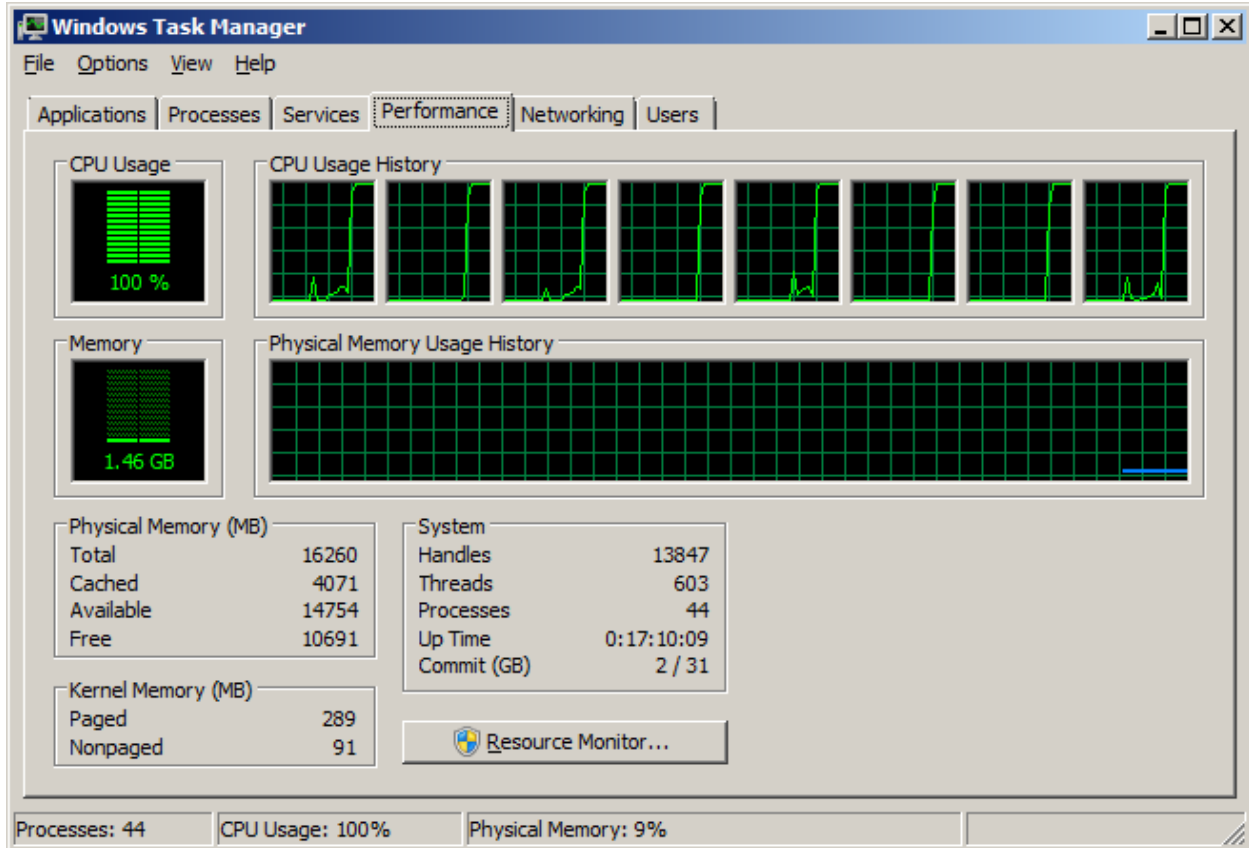


Figure 12 - Windows performance monitor.

This example used the **File Open File Set** shortcut to load all the necessary files in one operation, but the user could have done the same thing by loading the files via their individual menu items. For example, .msh files are opened via the menu item **File...Open Ascii Mesh File** (Figure 13).

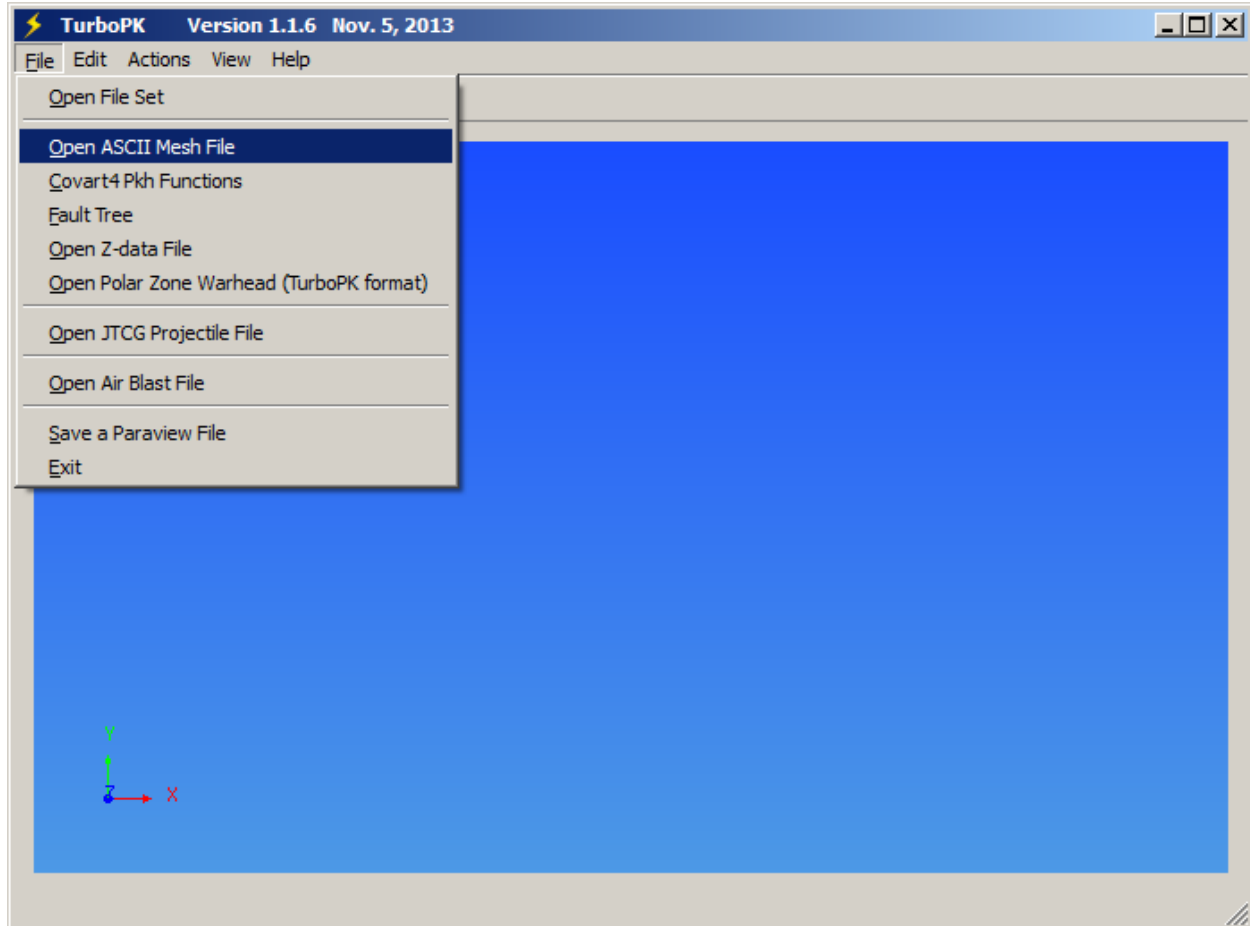


Figure 13 - Opening a mesh file.

The next sections describe the various **Action** menu items in the order of their appearance in the **Actions** menu.

3.0 Rayleigh Burst Point Option

This option is invoked via the menu item **Actions...Rayleigh Burst Points**. (NOTE: This option changed in 2014 to include the words "Target Coordinates", with an additional action for "Weapon Coordinates". This section discusses the Target Coordinate version.) Figure 14 shows this menu item being selected and the dialog box that pops up in response to it. This option generates a set of burst points in the target z-plane specified by the "Centroid Z" coordinate. The distribution center point in the x-y plane is specified by the "Centroid X" and "Centroid Y" coordinate values. "CEP" (Circle of Equal Probability). The equation for the cumulative distribution function for a Rayleigh distribution contains one parameter: σ . One CEP is 1.17σ .

$$F(x) = 1 - e^{-x^2/2\sigma^2}$$

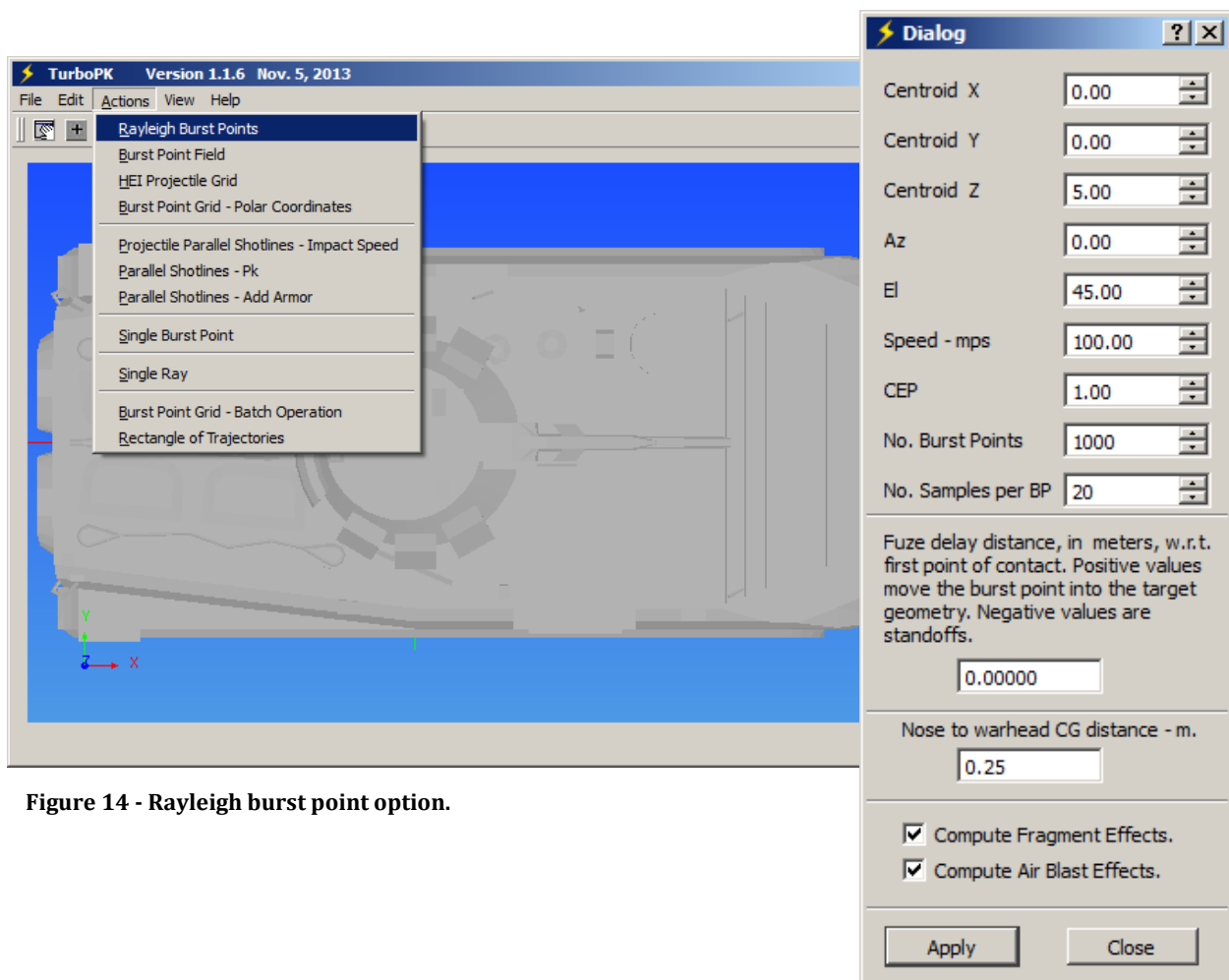


Figure 14 - Rayleigh burst point option.

"Az" and "El" are the azimuth and elevation angles of approach, respectively. Their conventions are as described in the previous section. "Number of Burst Points" and "No. Samples per BP" are self explanatory. "Fuze Delay Distance" applies to trajectories that intersect the actual target geometry. If a trajectory intersects the target geometry then a zero fuze delay distance makes the target intersection point the burst point for that trajectory. A positive delay distance moves the burst point "into" the target geometry while a negative delay distance backs the burst point "away" from the geometry model intersection point. Figure 15 and Figure 16 show the results of the Rayleigh Burst Point option for the parameters shown in Figure 14.

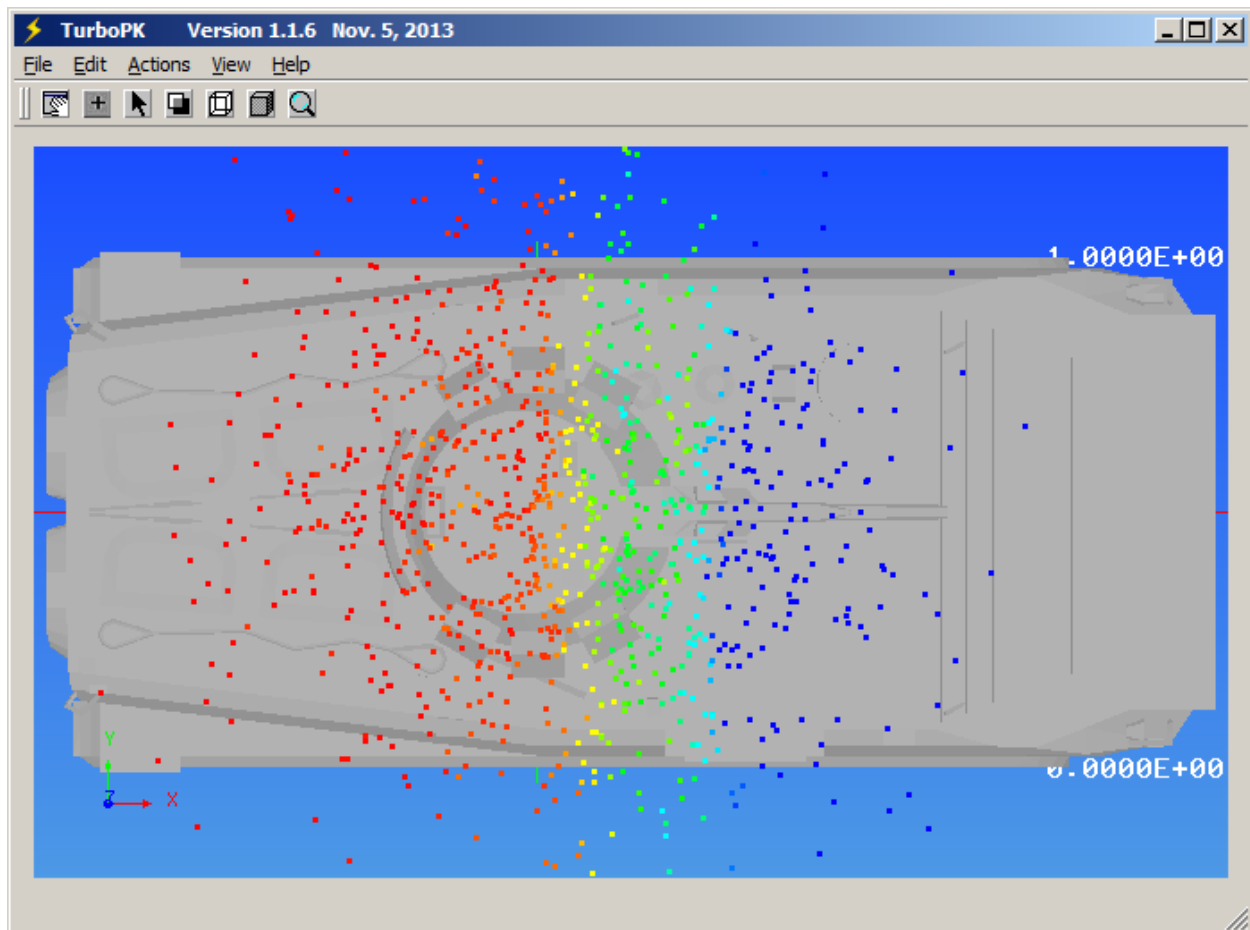


Figure 15 - Rayleigh Burst Point example, top view.

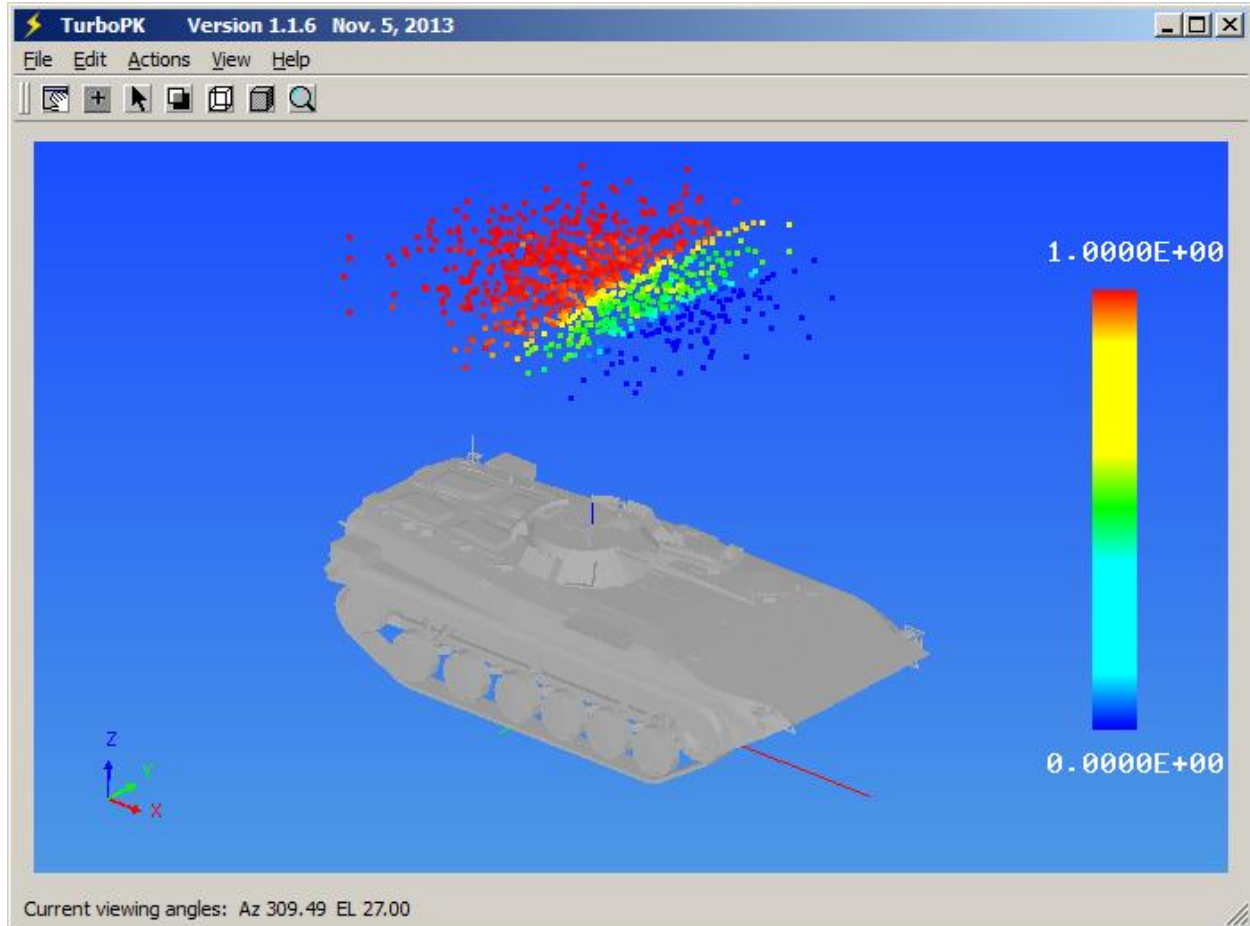


Figure 16 - Rayleigh Burst Point example, oblique view.

Figure 17 and-Figure 18 illustrate the effect of a negative delay distance. In this example the nominal burst point plane is the ground plane ($z = 0.0$ for this target model). Weapons are approaching from the front at a diving angle of 45 degrees. Figure 17 is for zero delay distance and Figure 18 if for negative 2-m delay distance.

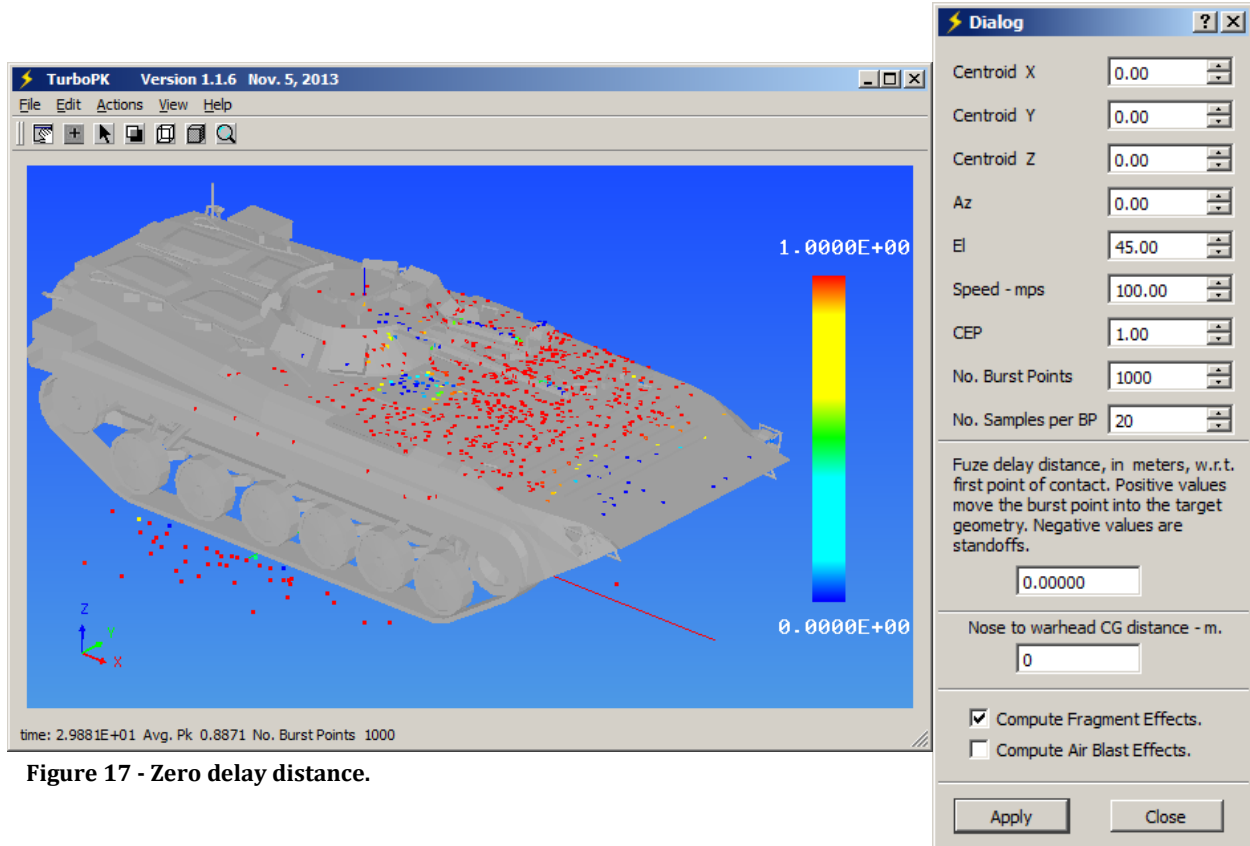


Figure 17 - Zero delay distance.

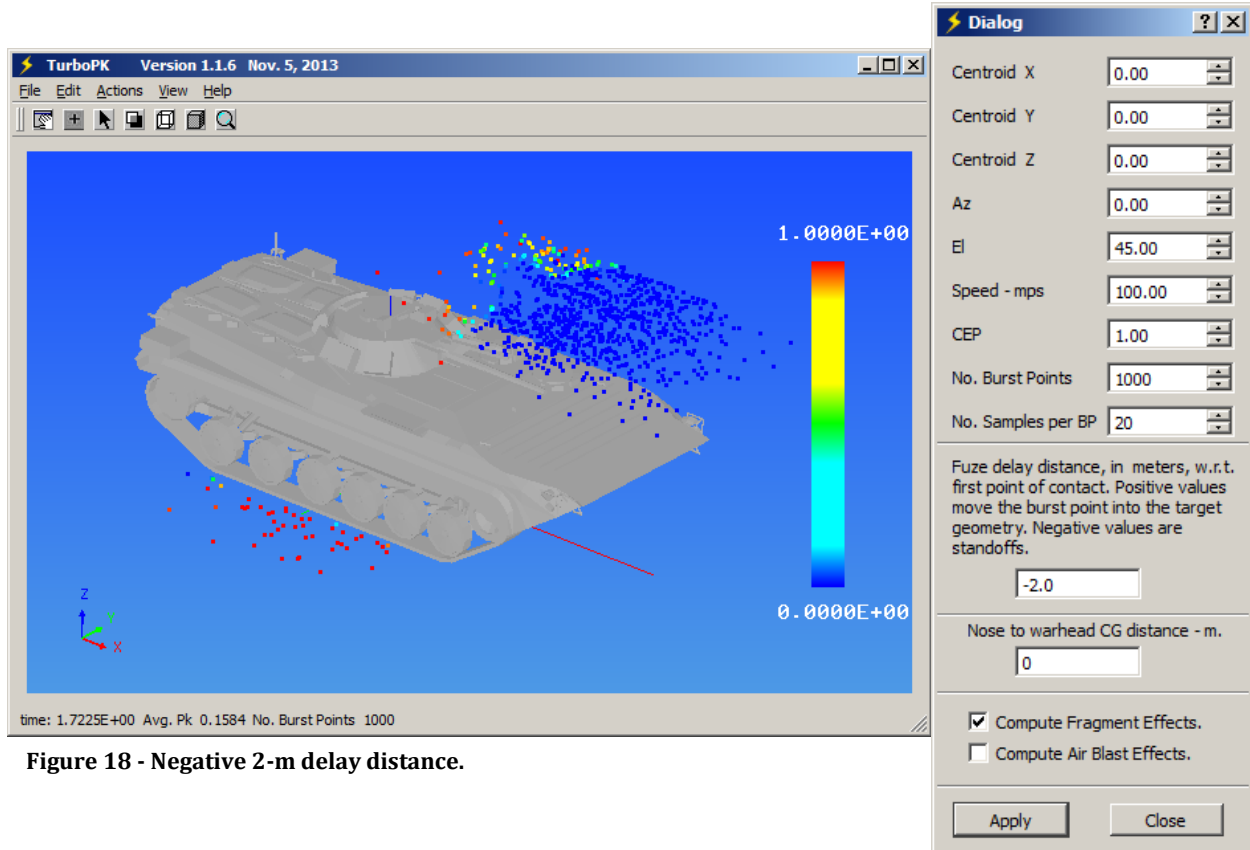


Figure 18 - Negative 2-m delay distance.

As indicated by the check boxes in Figure 17, only fragment effects were considered for the calculations shown in Figure 17 and Figure 18.

3.1 Rayleigh Distribution In Weapon Coordinates

The previous section allowed the user to define a Rayleigh distribution of burst points located in a fixed elevation relative to the target origin. Figure 19 shows an example distribution located 3-meters above the target origin. This capability is useful for analyzing artillery rounds which "fall" onto their targets.

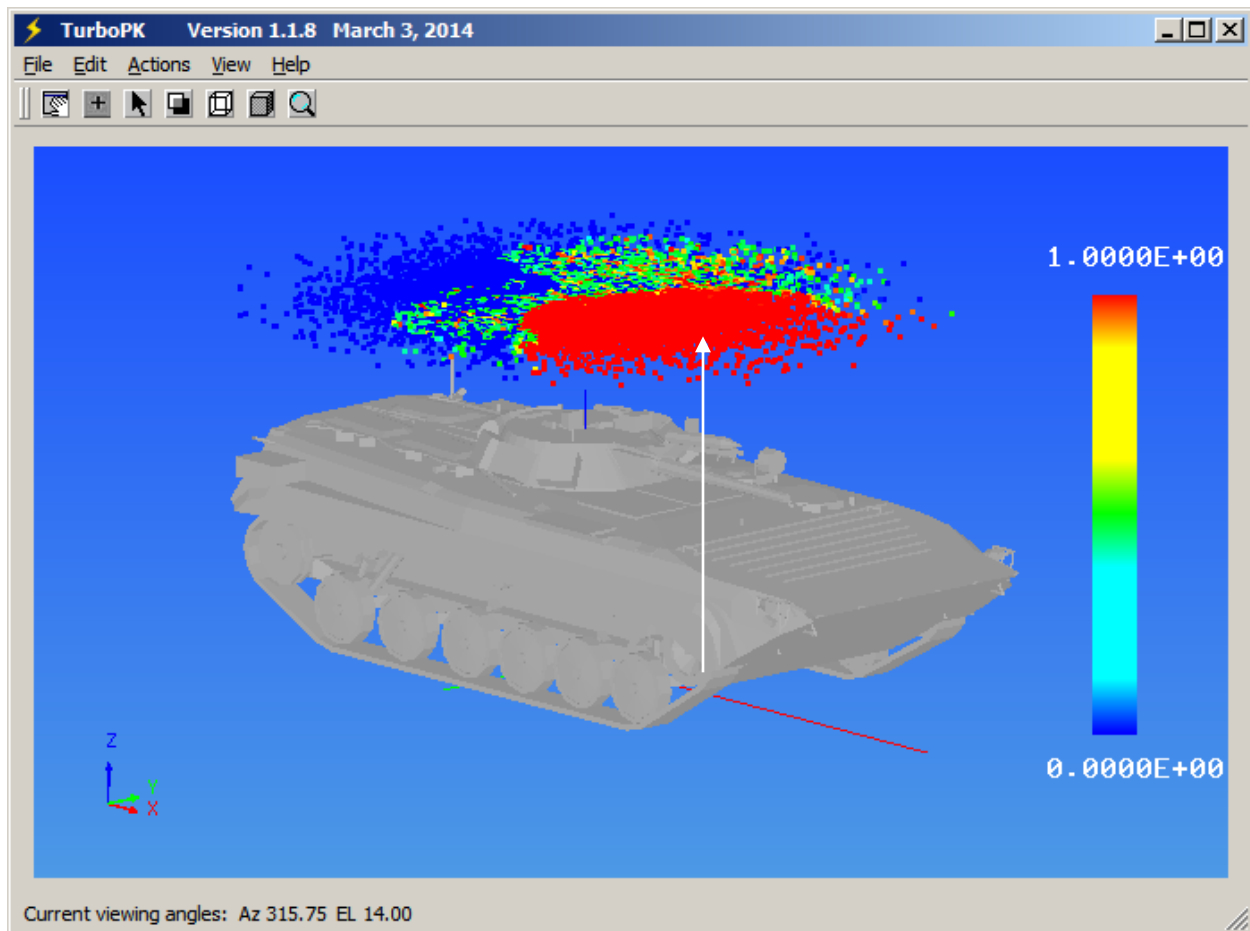


Figure 19 - Example Rayleigh distribution in target coordinates

For flat-fire weapons a Rayleigh distribution is also used to describe the miss distribution, but in this case the miss distribution is defined in a plane perpendicular to the projectile's direction vector, as opposed to being in the "ground plane" like it is for artillery projectiles. So a new option has been added to TurboPK allowing the user to specify a Rayleigh distribution for miss distances in weapon coordinates. Figure 20 shows the menu item for invoking this option.

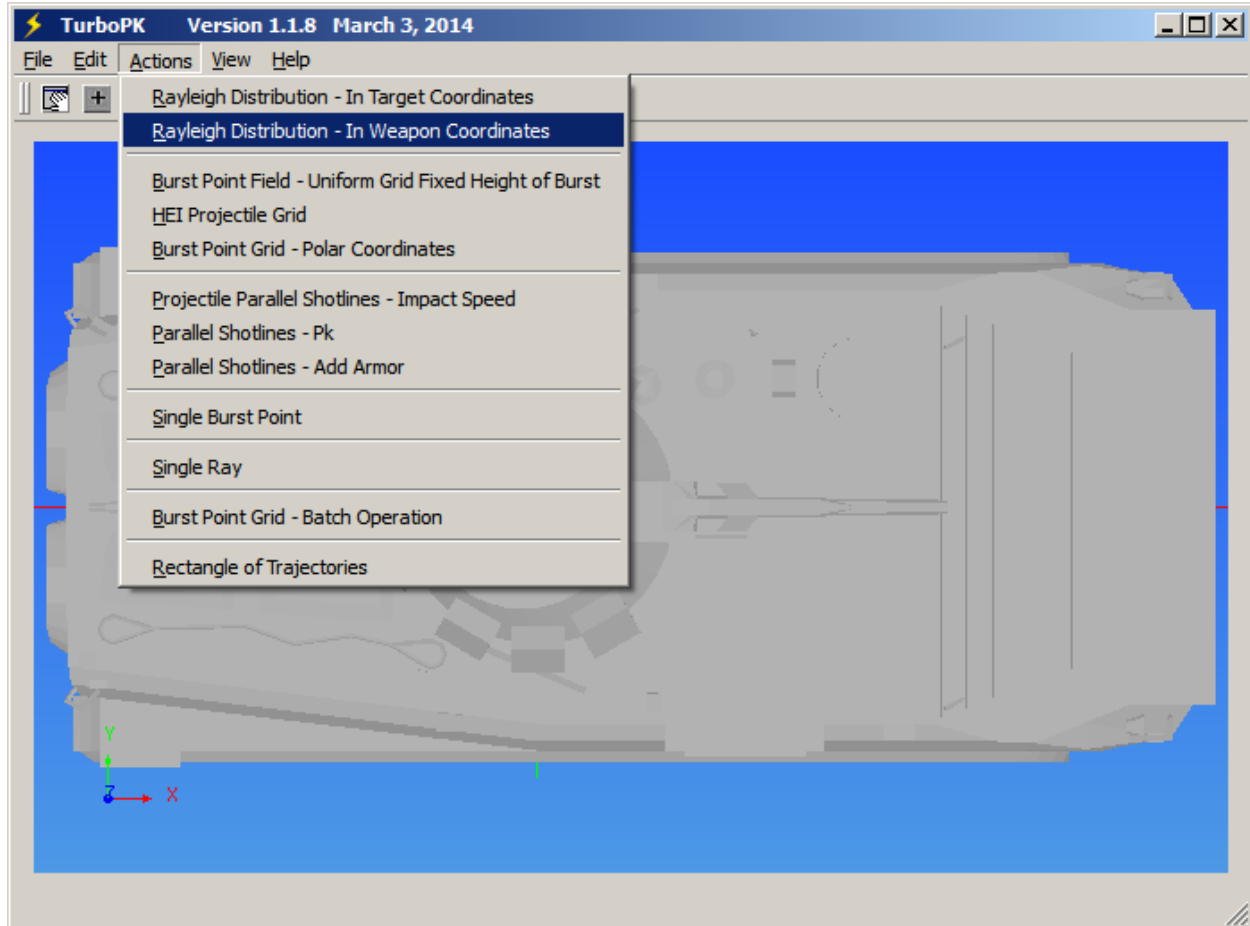


Figure 20 - Rayleigh Distribution in weapon coordinates option.

Figure 21 shows the dialog box that pops up in response to the menu item. This dialog box also presents a set of inputs for doing multiple weapons in a "salvo" but ignore those for the time being. Figure 23 shows the results of applying the parameters shown in Figure 21.

Rayleigh Distribution Parameters ? X

Az - deg.

El - deg.

Speed - mps

CEP - m.

Fuze Delay - m.

Nose to Warhead CG distance - m

No. Trajectories

No. MC Samples per trajectory

☐ Multiple Weapon Attack

Number per Salvo

Number of Salvos

No. MC Samples per round

Component Pks will be accumulated over all weapons prior to calculation of total target Pk.

Specify an aim point:

☒ Center of target bounding box.

☐ Specific target coordinates - m.

X Y Z

☒ Calculate frag effects.

☐ Calculate blast effects.

Apply Close

Figure 21 - Rayleigh Distribution / Weapon Coordinates Dialog

The Az/El pair (0.0, 0.0) specified in Figure 3 is head-on to the nose of the vehicle so the impact positions roughly describe a circle when viewed from the front of the vehicle.

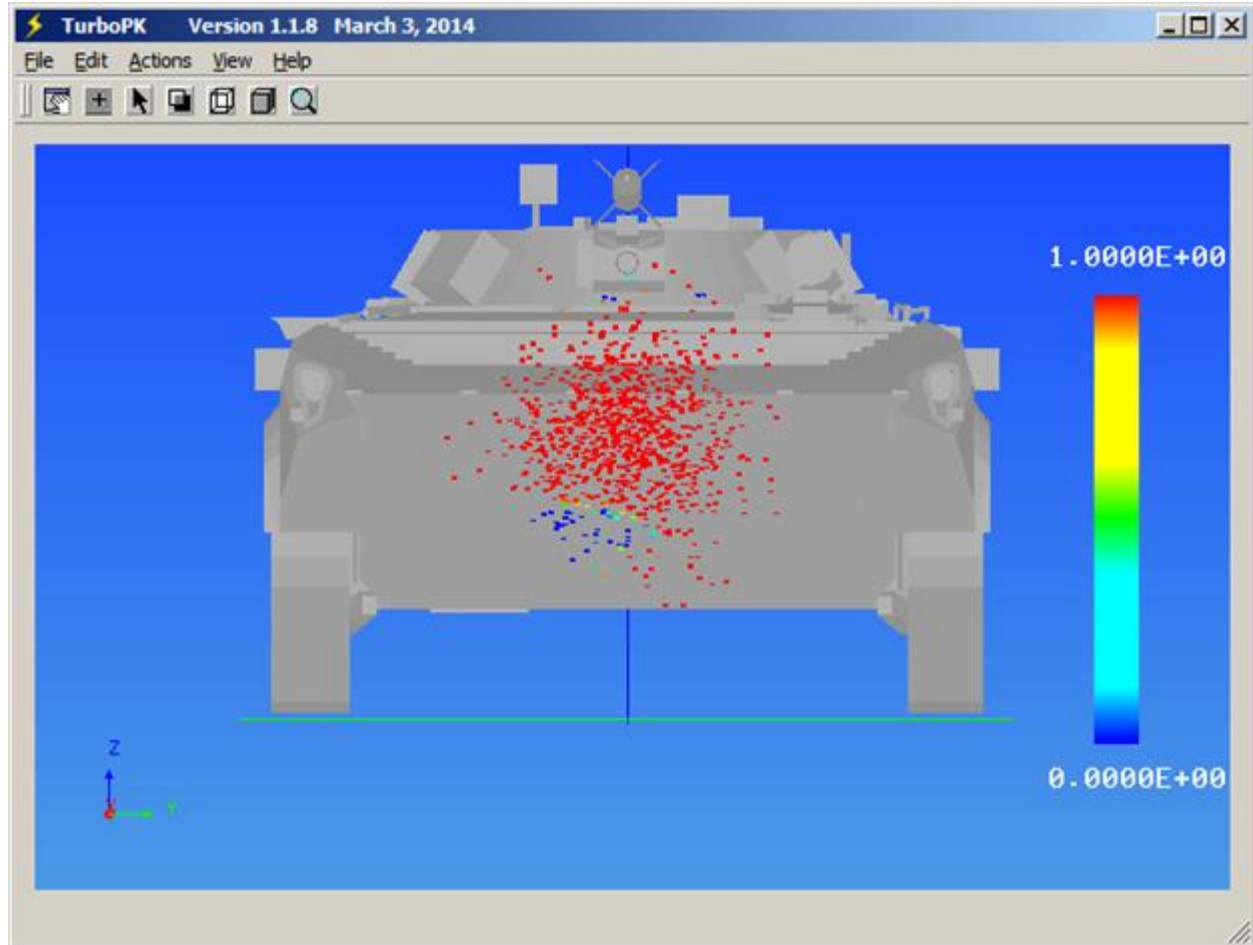


Figure 23 - Rayleigh Weapon Coordinates Results

Now consider the **Multiple Weapon Attack** option shown in the dialog box. To see its effect an example calculation will be done first with the single weapon parameters shown in Figure 24. That is, an attack at 70-degrees elevation angle, a CEP of 0.75 meters, and a nose-to-warhead distance of 0.5 meters. The latter means the warhead burst point will be 0.75 meters prior to the intersection of the projectile's trajectory with the target geometry. Figure 25 shows the distribution of burst points from a 70-degree elevation viewpoint, and Figure 26 shows the burst points from an oblique view. As indicated at the bottom of Figure 26 the average P_K over 1000 burst locations is 0.50 in this case. (Discussion continued after Figure 26.)

⚡
Rayleigh Distribution Parameters
?
✕

Az - deg.	<input type="text" value="0"/>
El - deg.	<input type="text" value="70"/>
Speed - mps	<input type="text" value="100"/>
CEP - m.	<input type="text" value="0.75"/>
Fuze Delay - m.	<input type="text" value="0"/>
Nose to Warhead CG distance - m	<input type="text" value="0.5"/>
No. Trajectories	<input type="text" value="100"/>
No. MC Samples per trajectory	<input type="text" value="20"/>

☐ Multiple Weapon Attack

Number per Salvo	<input type="text"/>
Number of Salvos	<input type="text"/>
No. MC Samples per round	<input type="text"/>

Component Pks will be accumulated
over all weapons prior to calculation of
total target Pk.

Specify an aim point:

☐ Center of target bounding box.

☒ Specific target coordinates - m.

X	Y	Z
<input type="text" value="2.5"/>	<input type="text" value="0"/>	<input type="text" value="0"/>

☒ Calculate frag effects.

☐ Calculate blast effects.

Figure 24 - Single Weapon Parameters

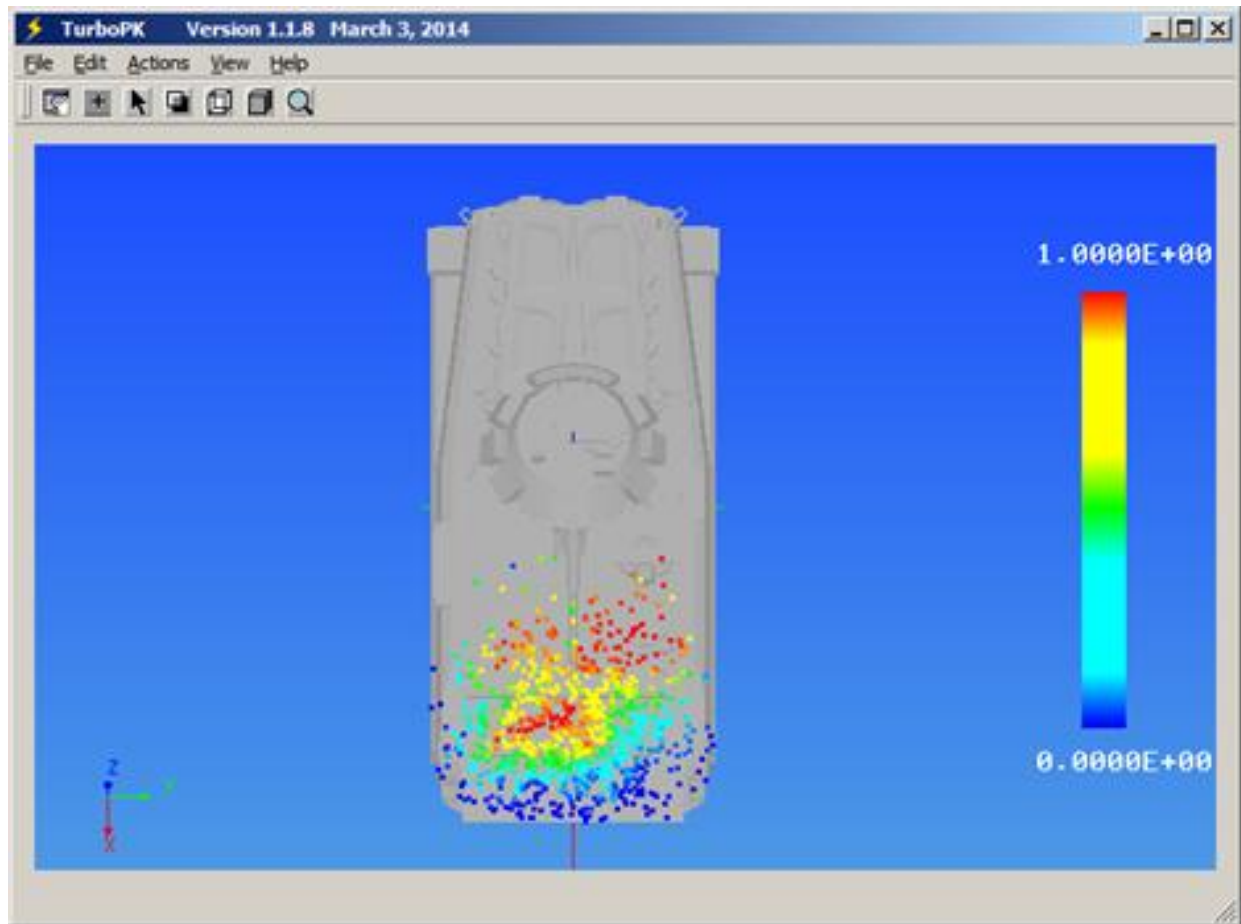


Figure 25 - 70 Degree Elevation

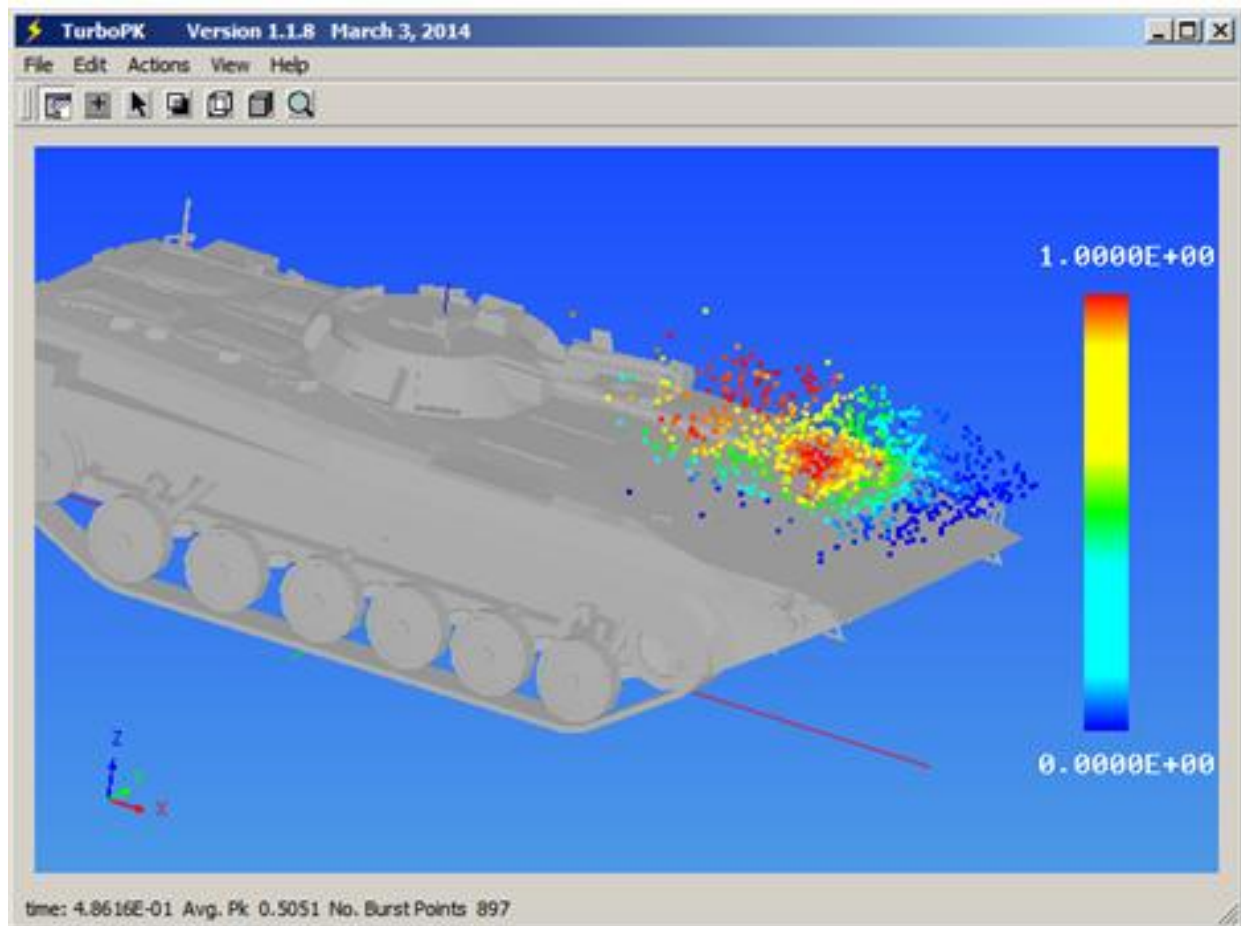


Figure 26 – Oblique View

Next, the **Multiple Weapon Attack** option is selected and the parameters set to the values shown in Figure 27. Each salvo consists of two projectiles. A total of 1000 salvos will be generated. For each salvo 20 Monte Carlo sample fragment patterns will be generated for each projectile in the salvo. Az, El, CEP, etc are as shown.

Rayleigh Distribution Parameters ? X

Az - deg.

El - deg.

Speed - mps

CEP - m.

Fuze Delay - m.

Nose to Warhead CG distance - m

No. Trajectories

No. MC Samples per trajectory

☒ Multiple Weapon Attack

Number per Salvo

Number of Salvos

No. MC Samples per round

Component Pks will be accumulated over all weapons prior to calculation of total target Pk.

Specify an aim point:

☐ Center of target bounding box.

☒ Specific target coordinates - m.

X Y Z

☒ Calculate frag effects.

☐ Calculate blast effects.

Apply Close

Figure 27 - Multiple Weapon Attack Parameters

In the case of the **Multiple Weapon Option** for each salvo the vulnerable component P_K values are accumulated over all projectiles in a salvo prior to fault tree evaluation. Figure 28 shows the post-calculation results in the status bar at the bottom of the main window. In this example the average salvo- P_K is estimated to be 0.9316. There is no burst point P_K display because the target P_K values are accumulated over all burst locations in each salvo.

Note that simply compounding the single round average P_K for two rounds would give a compounded average P_K of 0.75.

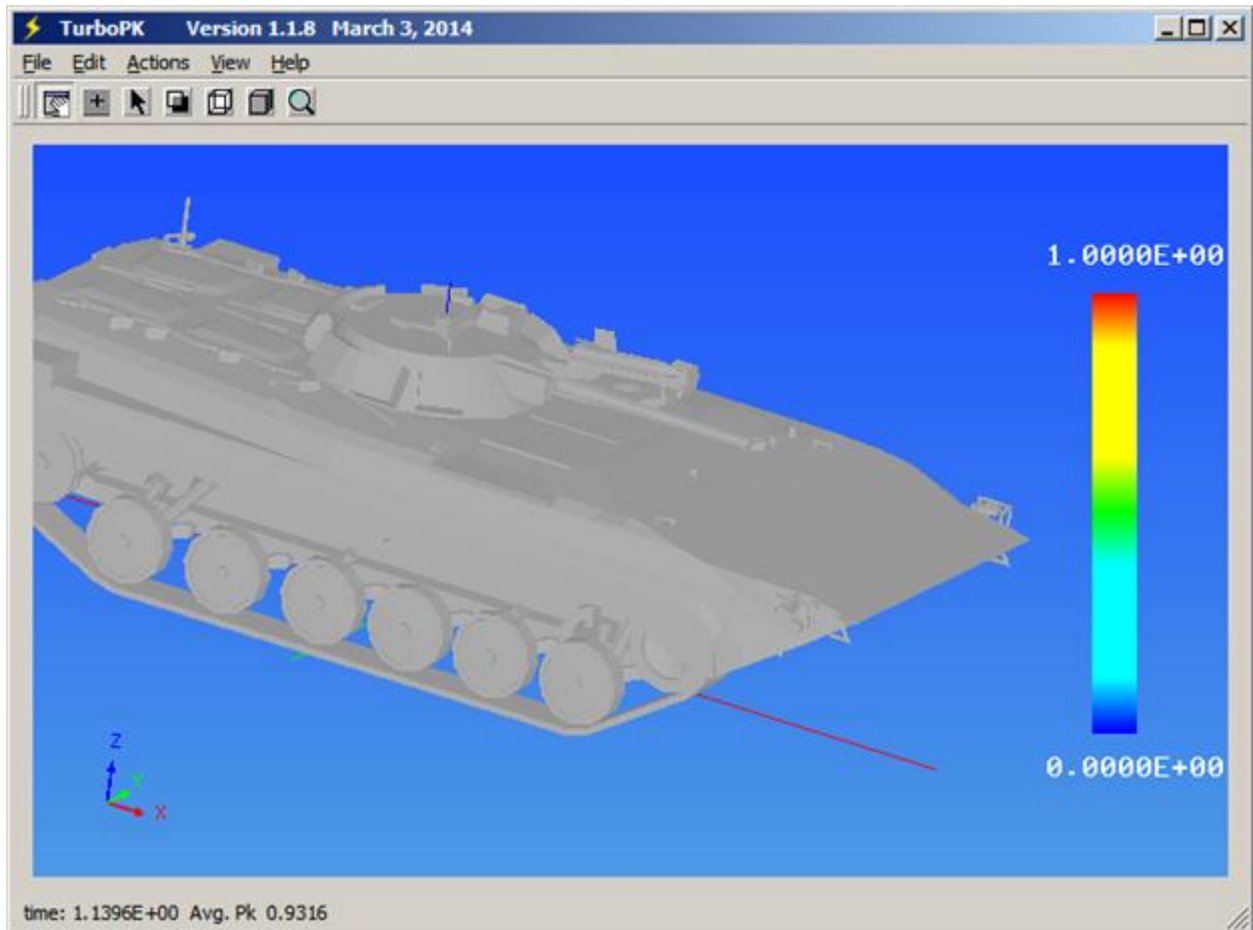


Figure 28 - Multiple Weapon Attack Results

4.0 HEI Projectile Grid

This option simulates the fragmentation effects of High Explosive Incendiary projectiles. It is accessed via the **Actions...HEI Projectile Grid** menu item. Figure 29 shows this menu item and the dialog box that pops up in response to it. Azimuth and Elevation angles obey the conventions previously described. This option creates a grid of projectile trajectories that overlays the presented area of the target from the specified azimuth and elevation angles. "Spacing" refers to the spacing between projectile shotlines. "Delay Distance" is a fuzing delay distance as described in the previous sections. "No. Samples" is the number of Monte Carlo point-burst samples done at each projectile burst location.

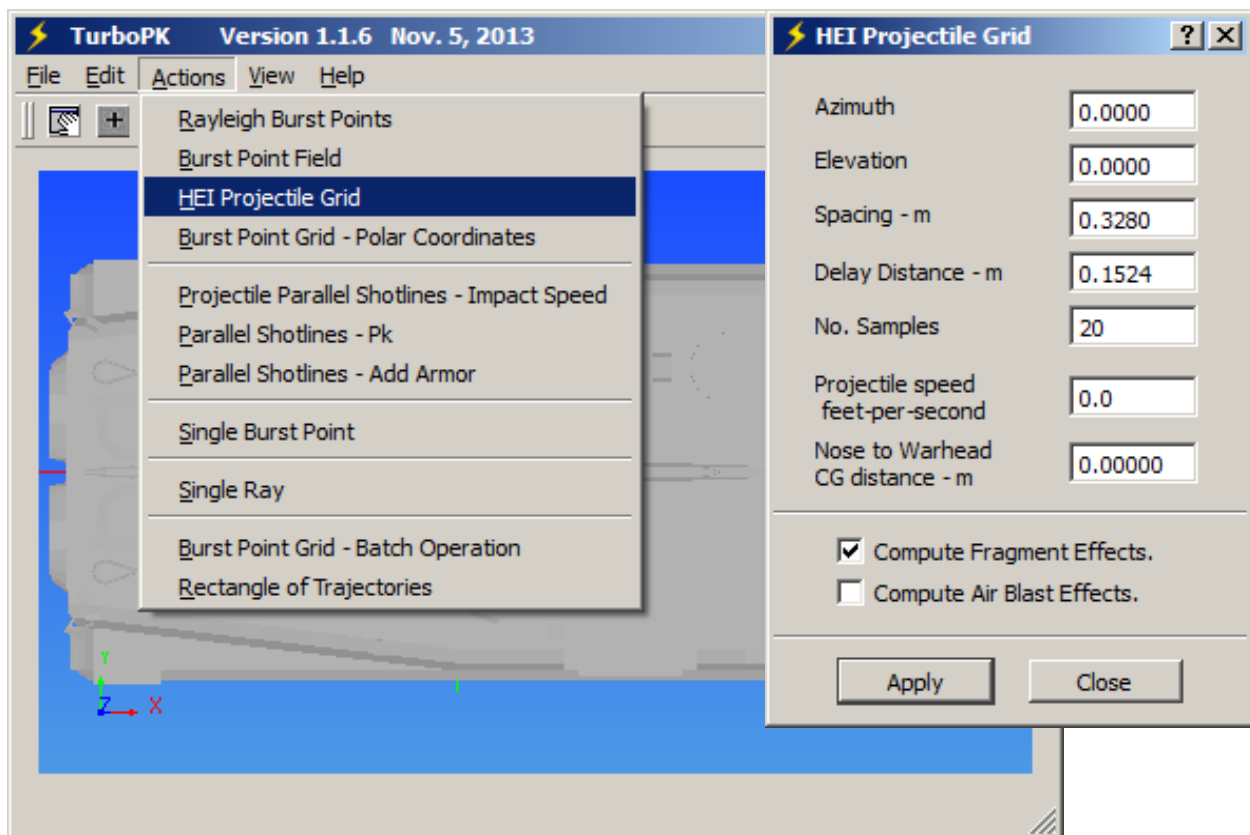


Figure 29 - HEI projectile grid option.

By default the fuzing delay distance is set to 0.1524-m (6.0 inches), the "projectile" will be placed 6-inches inside the geometry model relative to its first intersection with the geometric model. Changing the Elevation angle to 90.0 degrees and clicking on the Ok button should produce something like Figure 30.

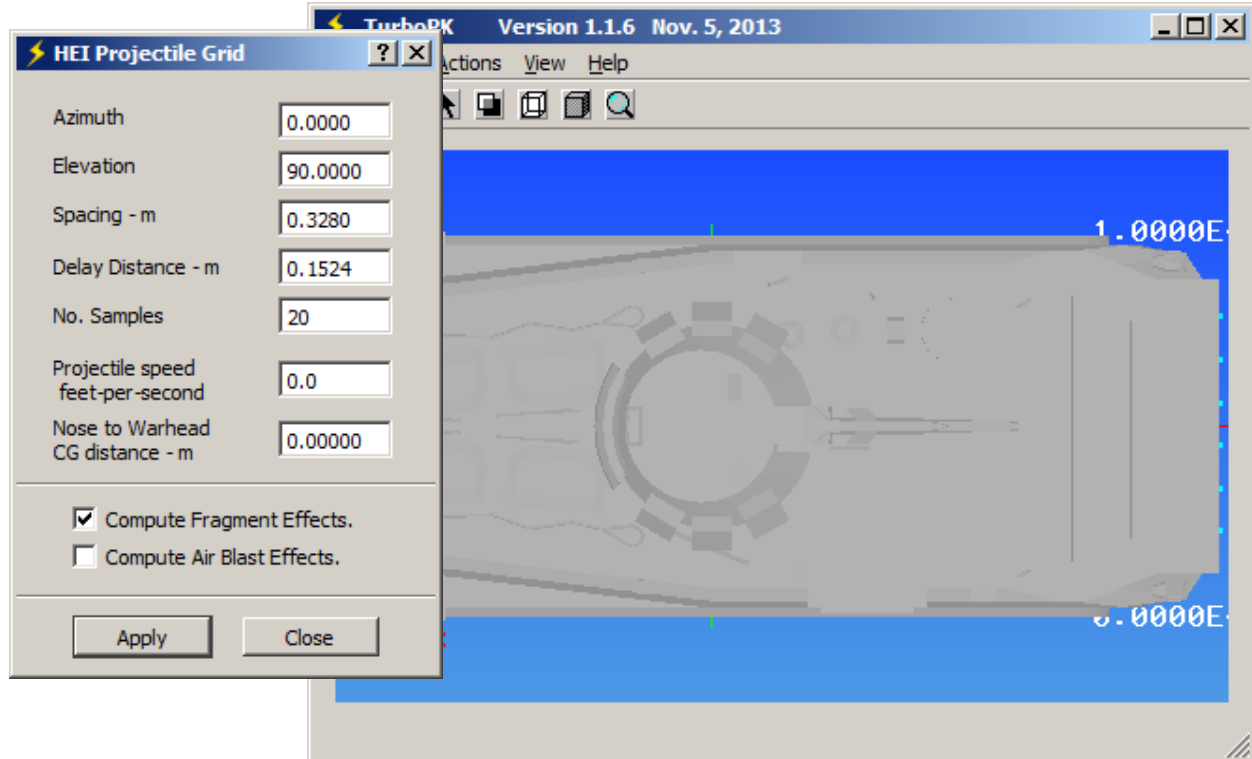


Figure 30 - HEI example.

The P_K markers in this case are drawn at the burst points, which in this case are internal to the geometric model, hence not visible. To make them visible click on the transparency control button (fourth from the left) and set the transparency to 40%. The P_K markers are now visible (Figure 31).

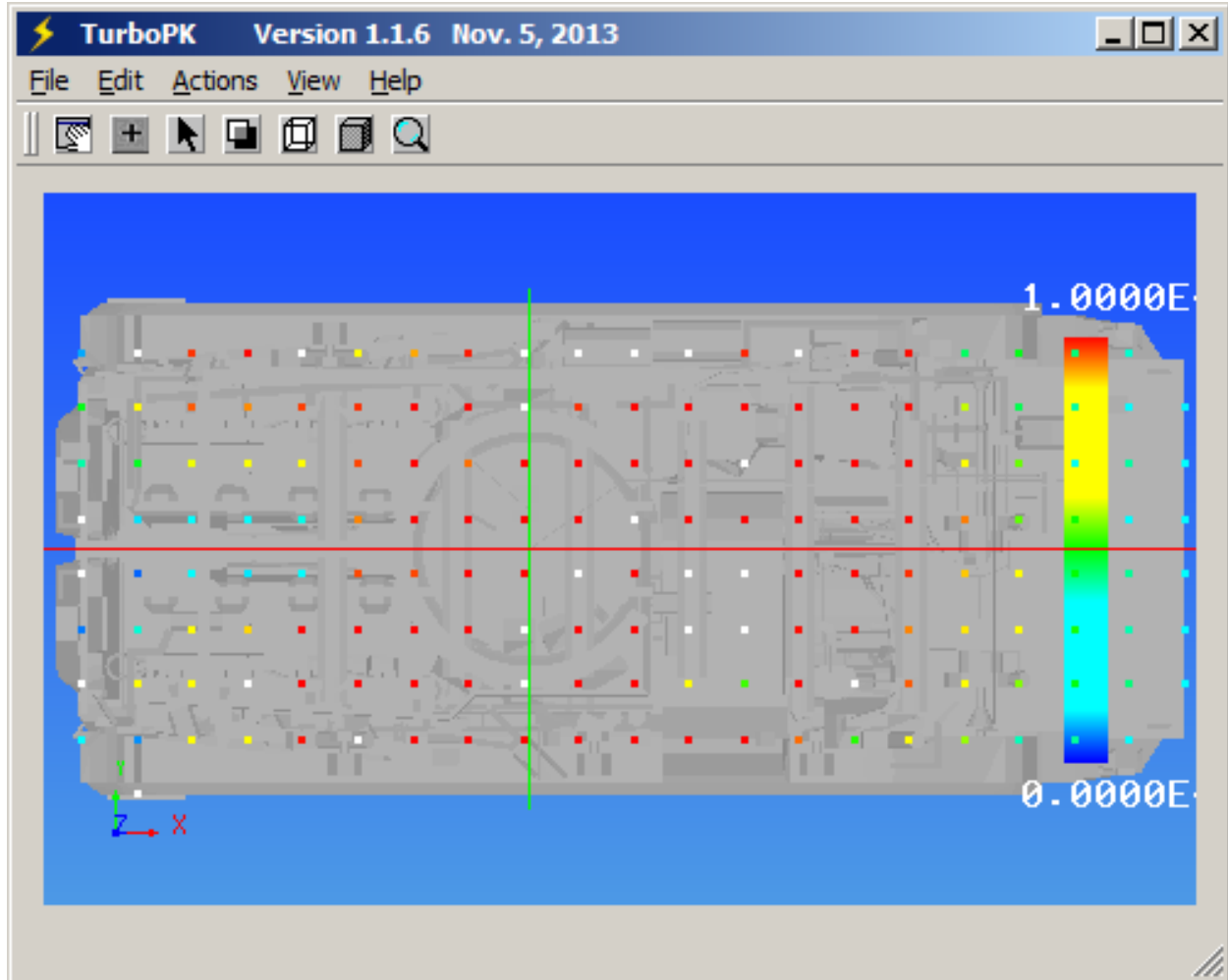


Figure 31 - HEI example, transparent view.

Note the white P_K markers in Figure 31. These are associated with burst points that are located *inside* a volume-mode (solid) object. It is not clear how to proceed for these burst points because the physical state of the projectile is called into question for these cases, and the characterization of the fragmentation (mass, speed, direction) are for a projectile in free air, not one buried inside a solid object of some kind. Therefore, TurboPK ignores burst points that are inside solid objects.

5.0 Projectile Parallel Shotline Options

One of the standard techniques in vulnerability analysis is to analyze an entire "grid" of parallel shotlines. The grid overlays the projection of the target model from whatever (azimuth, elevation) pair a user specifies. Figure 32 for example displays a shotline grid overlaying the example vehicle model from the angles (135,0). Blue lines have been added to delineate grid cells and red dots have been added to indicate shotline locations.

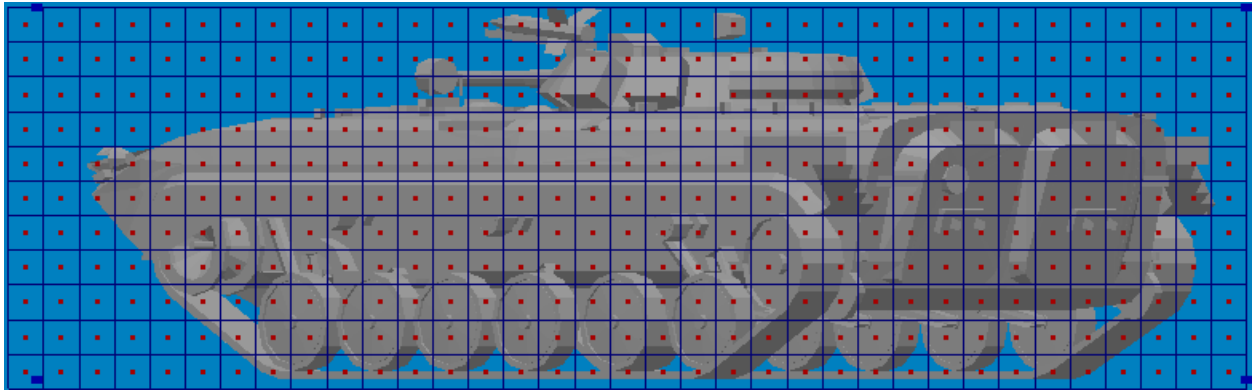


Figure 32 - Example shotline grid.

The first parallel shotline option is accessed via the menu item **Actions...Projectile Parallel Shotlines - Impact Speed** (Figure 33).

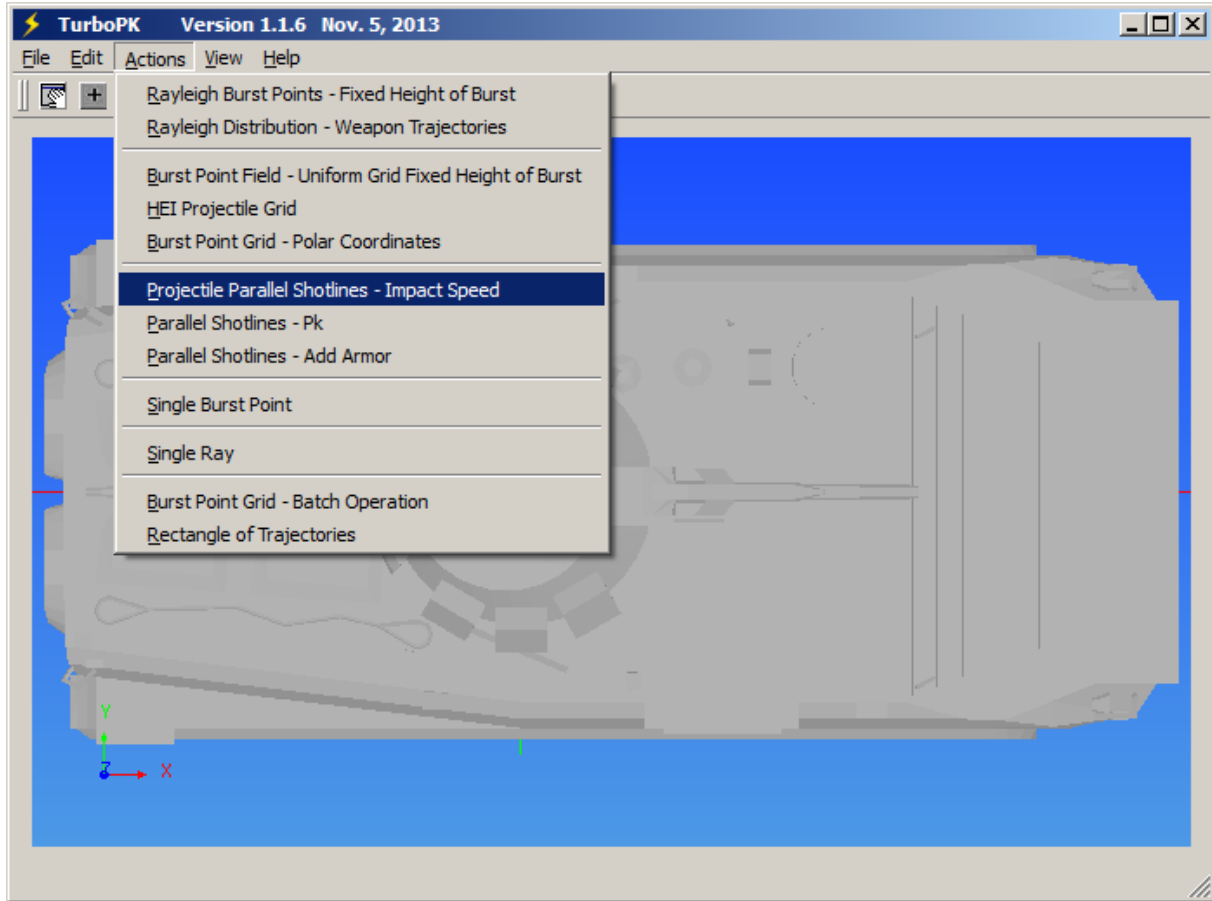


Figure 33 - Parallel shotline option.

Selecting **Actions...Projectile Parallel Shotlines - Impact Speed** pops up the dialog box shown in Figure 34. The parameters **Azimuth**, **Elevation**, and "**Shotline Spacing**" are as described previously. **Speed** refers to the projectile, or fragment, impact speed. The center section of the dialog is a set of radio buttons for electing to analyze a projectile, or a cube shaped steel fragment, or a spherical steel fragment. If **Use Projectile** is selected TurboPK uses whatever projectile definition was last loaded by the user. If **Use Cube Fragment - Steel** or **User Sphere Fragment - Steel** is selected then the **Frag Mass - Grains** window is activated. The input control labeled **Max RHA Addition. inches** is described in a later section of this document.

Parallel Shotlines ? X

Azimuth - deg 0.00

Elevation - deg 0.00

Spacing - in 1.00

Speed - fps 2700.00

☒ Use Projectile
☐ Use Cube Fragment - Steel
☐ Use Sphere Fragment - Steel

Frag Mass - grains 60.00

Max RHA Addition, inches 2.00

Apply Close

Figure 34 - Parallel shotlines dialog box.

Changing the **Elevation** parameter to 90 degrees and clicking on the **Apply** button should produce an image similar to Figure 35. The colors represent the projectile's speed at intersection with the first vulnerable component encountered along its shotline. Spin buttons in the dialog box allow for easy modification of azimuth, elevation, shotline spacing, and impact speed.



Figure 35 - Projectile shotline grid example.

The second parallel shotline option is accessed via the menu item **Actions...Parallel Shotlines - Pk**. Figure 36 shows its effect for 90 degrees elevation and the projectile threat.

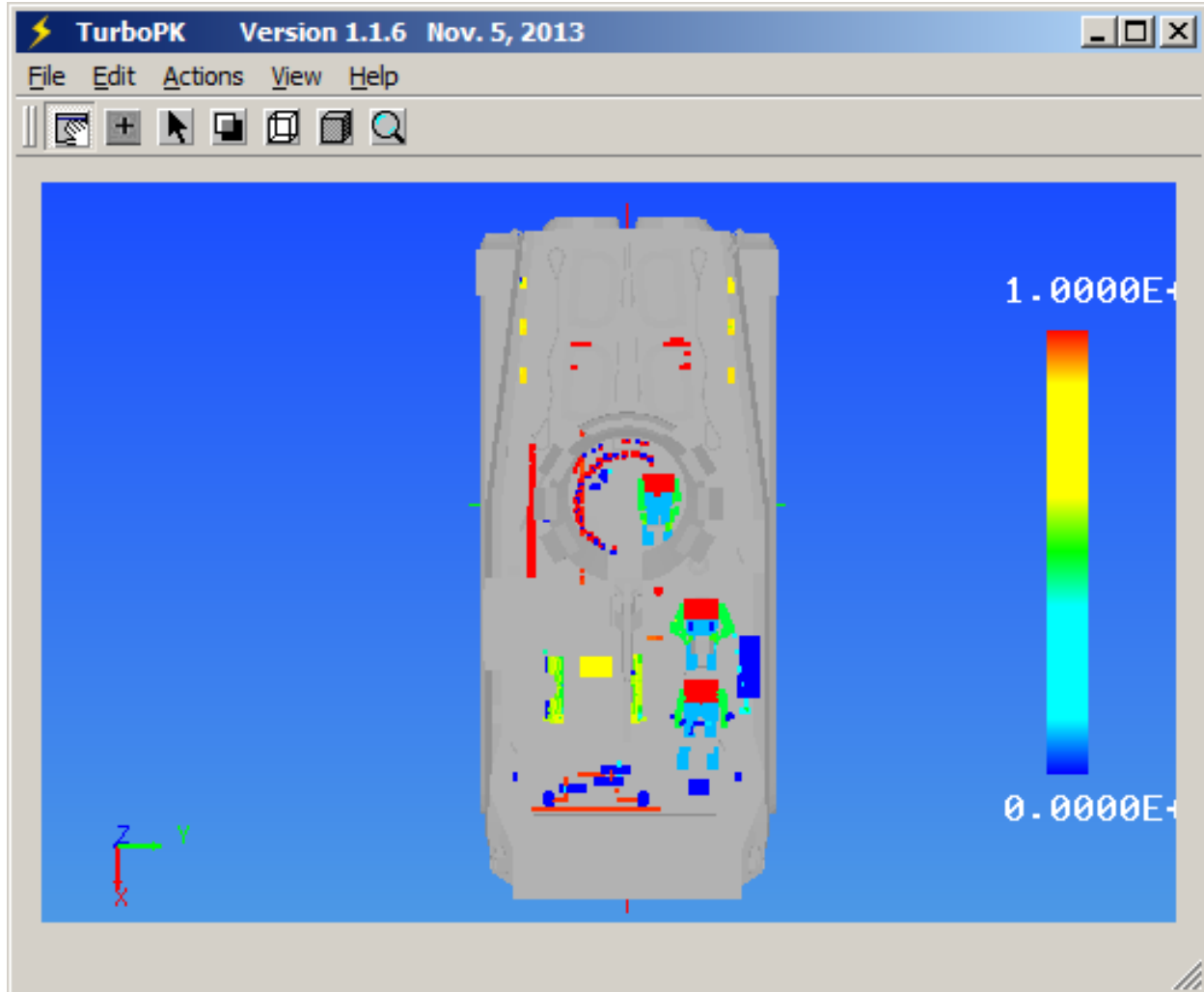


Figure 36 - Shotline Pk example.

Note that the color scale is now from 0.0 to 1.0. This option will not work unless a set of vulnerable component $P_{K/H}$ (probability-of-kill-given-a-hit) functions have been loaded for the target model. $P_{K/H}$ functions specify a damage probability for a component as a function of the mass and speed of the object striking them. Most commonly, $P_{K/H}$ functions are expressed as bi-variate (mass and velocity) piecewise-linear functions. When a user invokes the menu item **Actions...Parallel Shotlines - P_K** the code will compute the probability of killing one or more vulnerable components found along a shotline. For a given shotline it does that by applying the appropriate $P_{K/H}$ function for each vulnerable component encountered along the shotline. This results in a sequence of P_K values $\{P_{K1}, P_{K2}, \dots, P_{Kn}\}$ for the vulnerable components along the shotline in question. The total probability of killing one or more components is then computed according to the so called survival rule:

$$P_K = 1.0 - \prod_{i=0}^n (1.0 - P_{K_i})$$

It is these total P_K values that are color-coded as markers for each shotline.

6.0 Parallel Shotlines - Add Armor Option

This option is accessed via the menu item **Actions...Parallel Shotlines - Add Armor** (Figure 37). This pops up the dialog shown in Figure 38

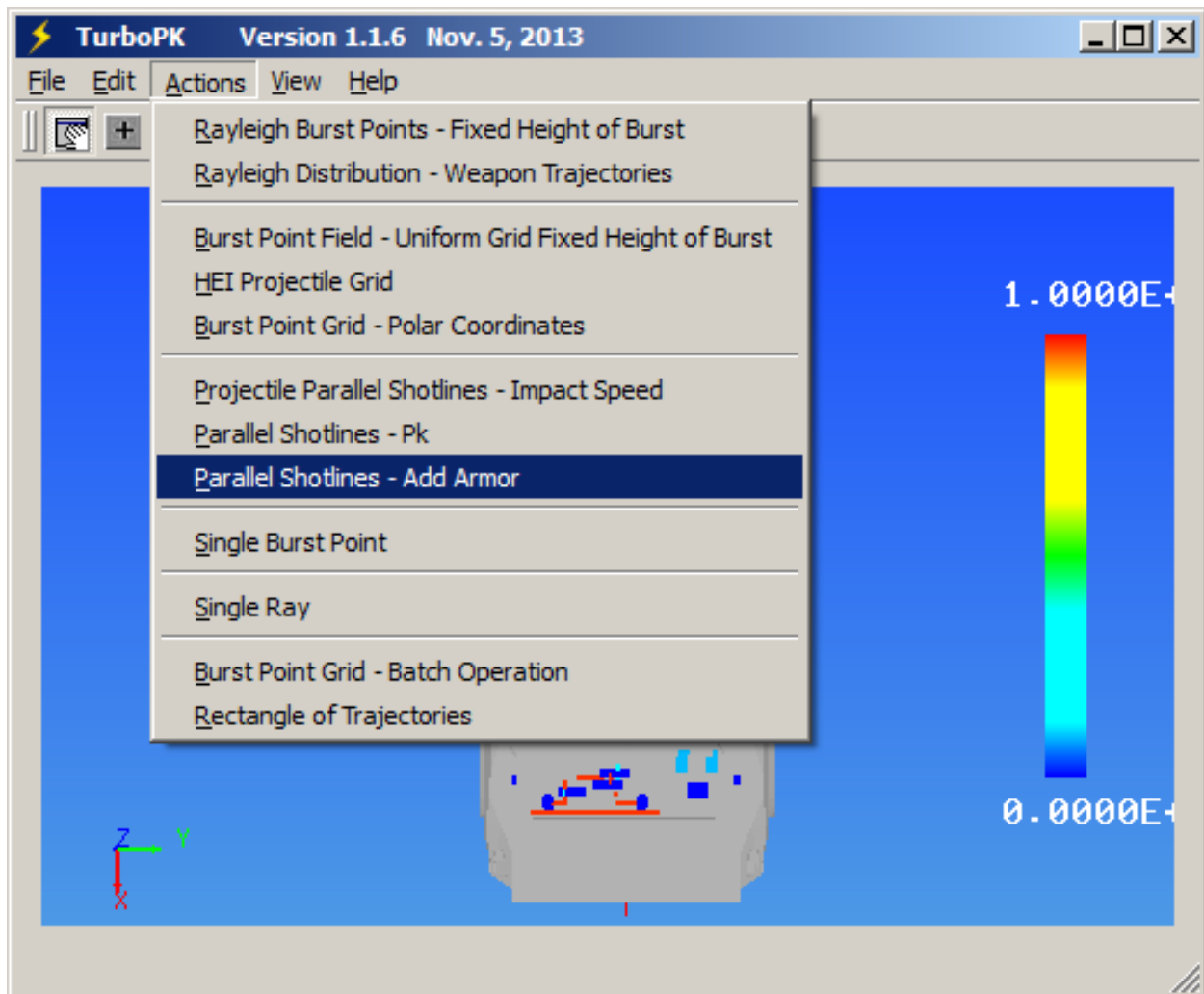


Figure 37 - Add Armor option.

This is the same dialog as for the other parallel shotline options, but in this case TurboPK will do two calculations. First, it will determine the projectile (or fragment) impact mass and speed at the first vulnerable component encountered along each shotline. Second, for those shotlines where the impact speed and mass are greater than zero it will determine how much additional RHA (rolled homogeneous armor) is required to stop the projectile (or fragment). It does not add the RHA to the geometry of the target model, it just adds it to

the list of material segments along the shotline in question. Figure 39 shows the result of doing this for the dialog parameters shown in Figure 38.

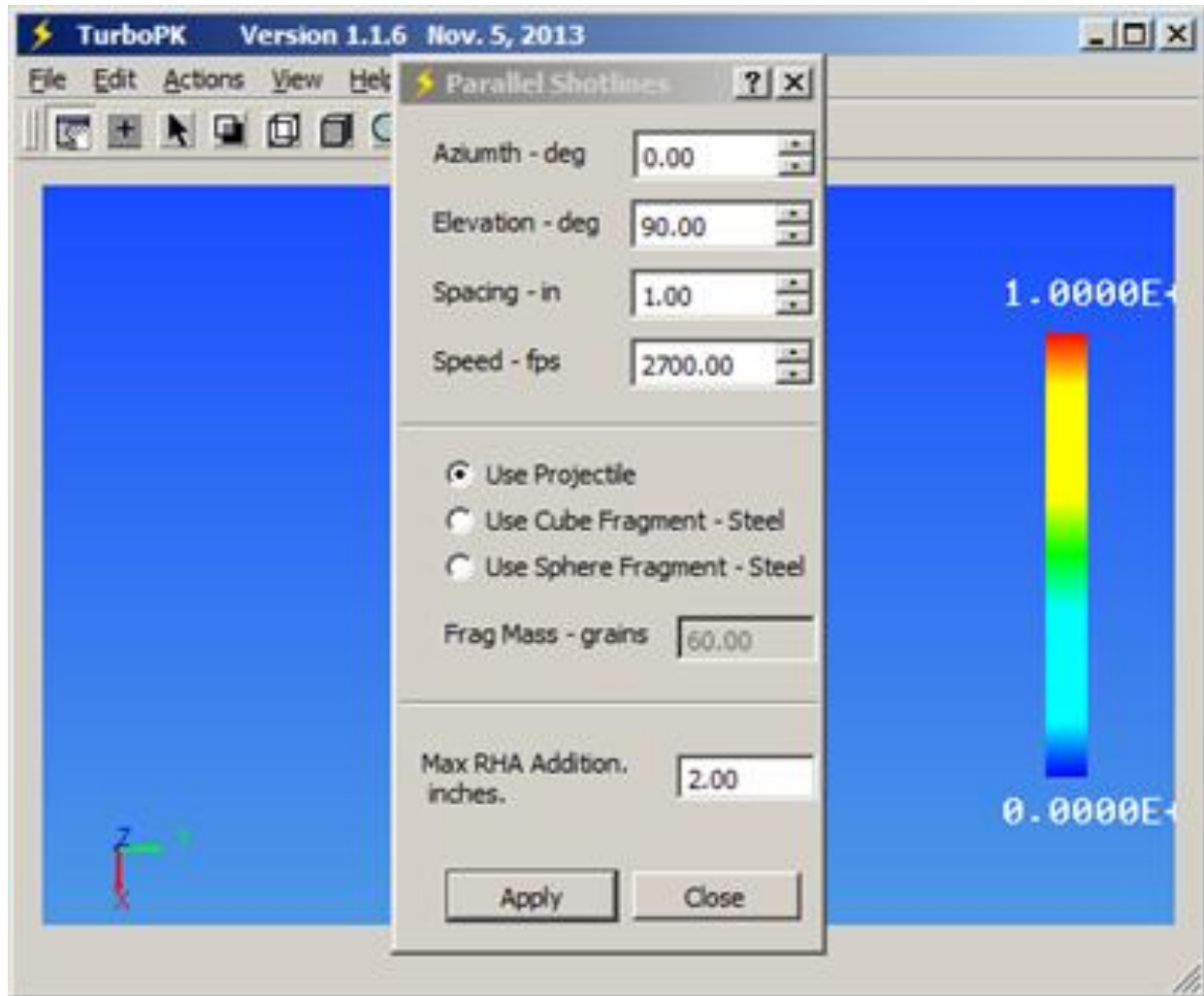


Figure 38 - Add Armor dialog.

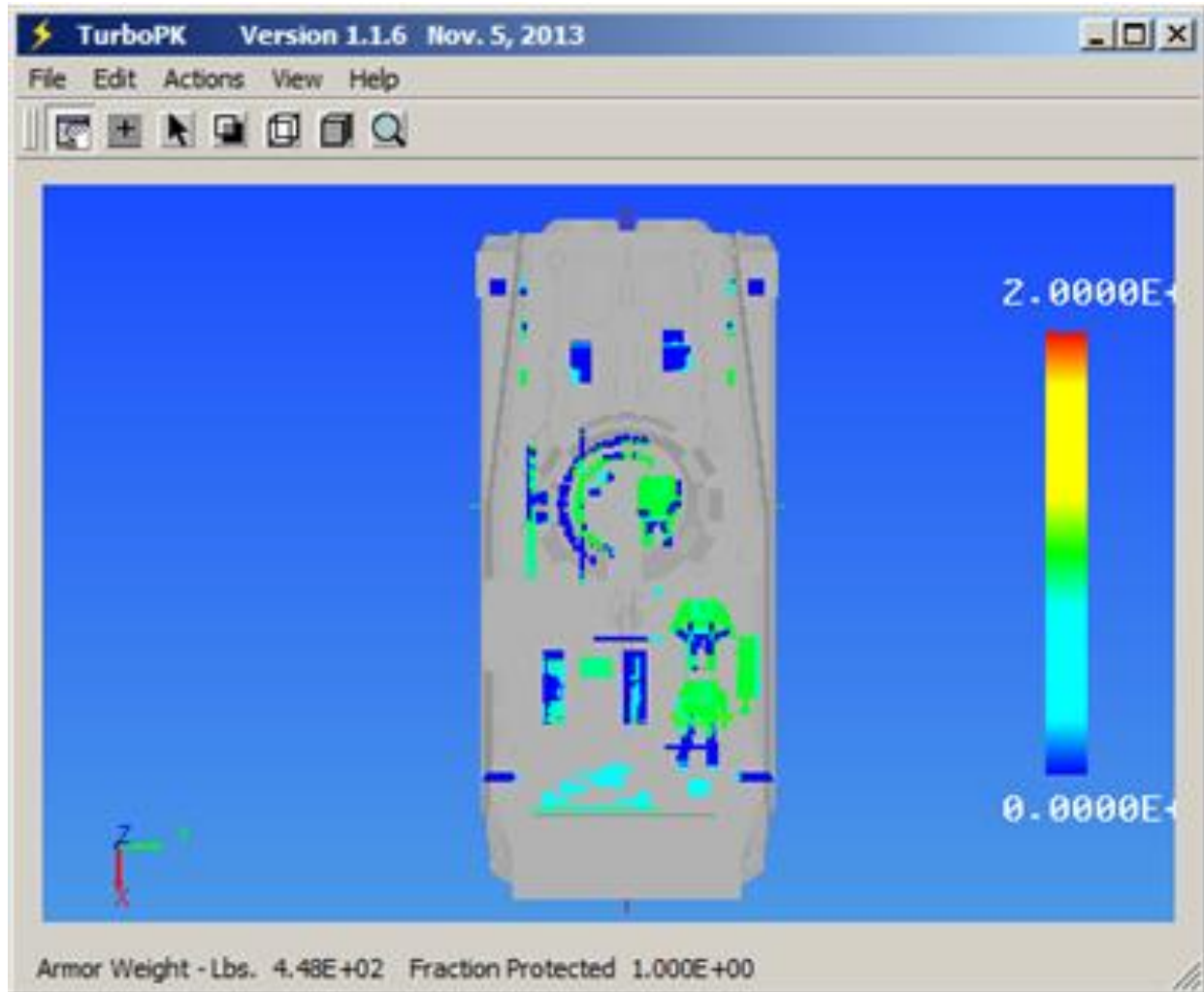


Figure 39 - Add Armor example calculation.

In Figure 38 note the scale is now 0.0 to 2.0 (inches of RHA). So the parameter **Max RHA Addition. inches** is the maximum that TurboPK will apply for the calculation in question. Note also the text in the status bar at the bottom of the main window. This text states that the weight of RHA added was roughly 448 pounds, and that 100% of the shotlines were protected. So no shotline required more than 2.0-inches of RHA to stop the projectile.

The effect of reducing the maximum RHA addition to 0.75-inches along a shotline is shown in Figure 40. A black shotline marker means more than 0.75-inches RHA are required along that shotline. Restricting the add-on RHA to no more than 0.75 inches reduces the total additional RHA weight to 103-lbs, but only 54.5% of the shotlines are now protected.

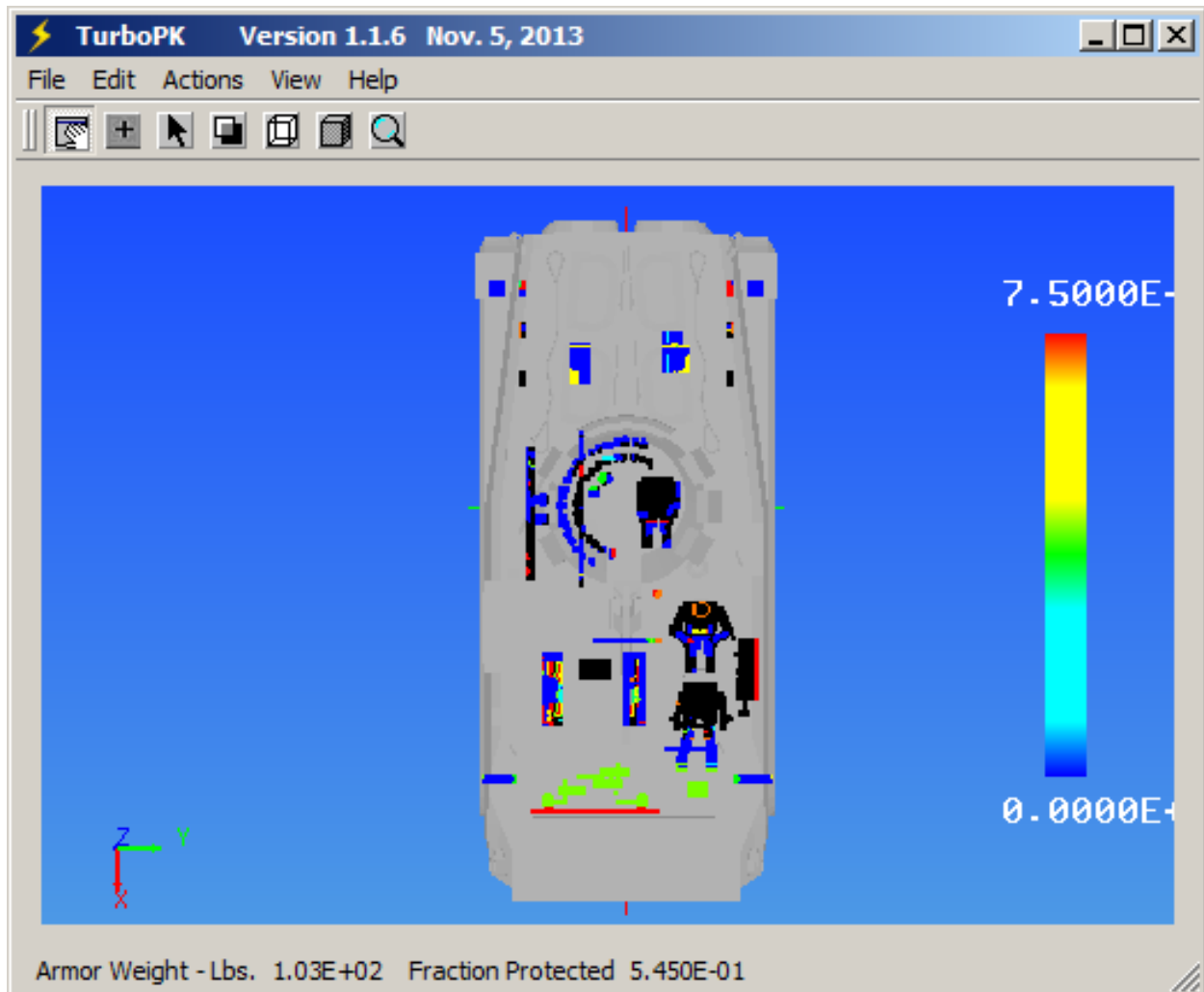


Figure 40 - Reducing the RHA add-on maximum to 0.75-in.

7.0 Single Burst Point Option

This option is invoked from the menu item **Actions...Single Burst Point**. In response the dialog box shown in Figure 41 pops up with the usual set of parameters. In Figure 41 the viewpoint has been changed to be almost side on. Figure 42 shows the fragment rays emanating from the example burst point. A white arrow was added to indicate the weapon approach angle.

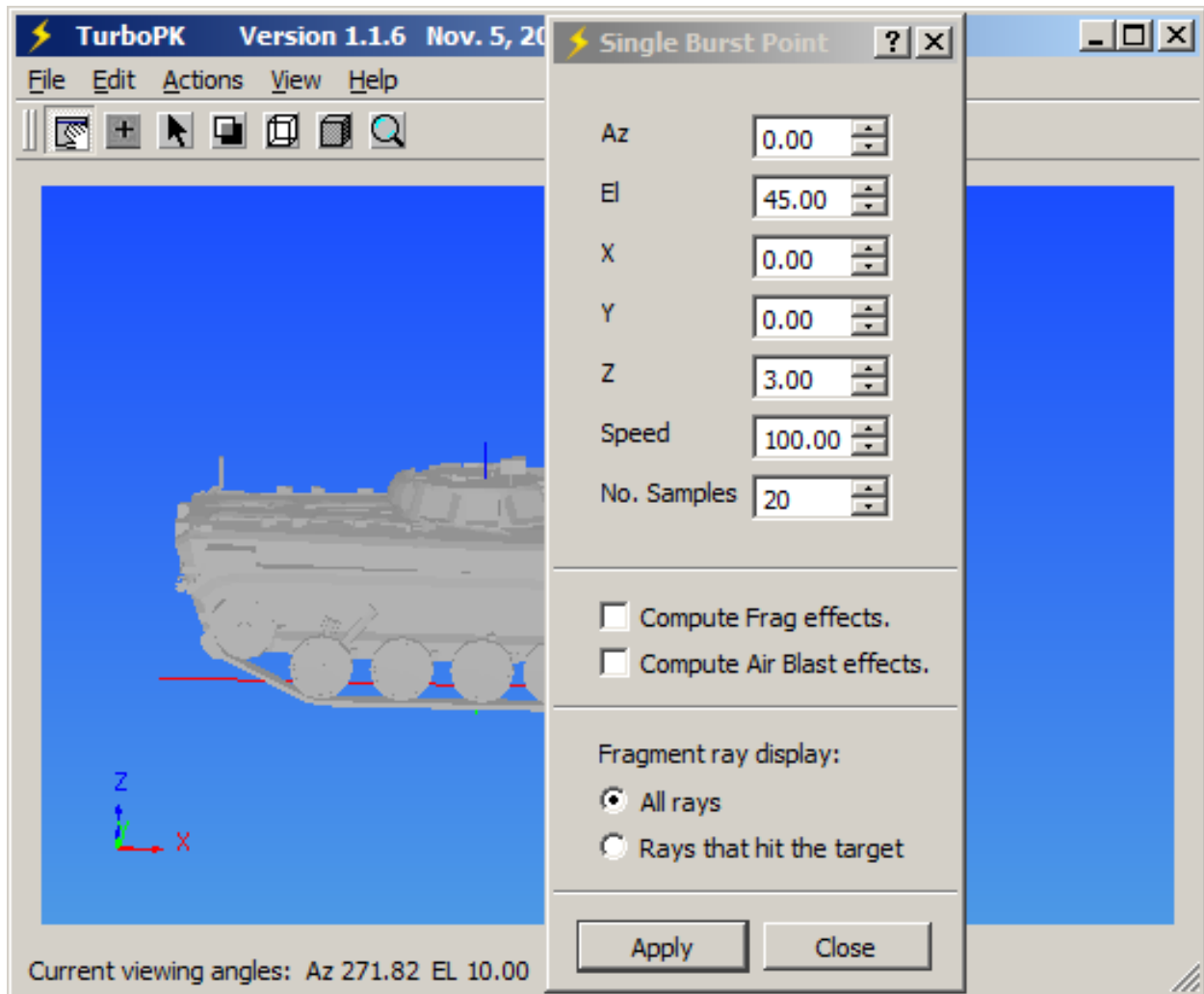


Figure 41 - Single Burst Point dialog.

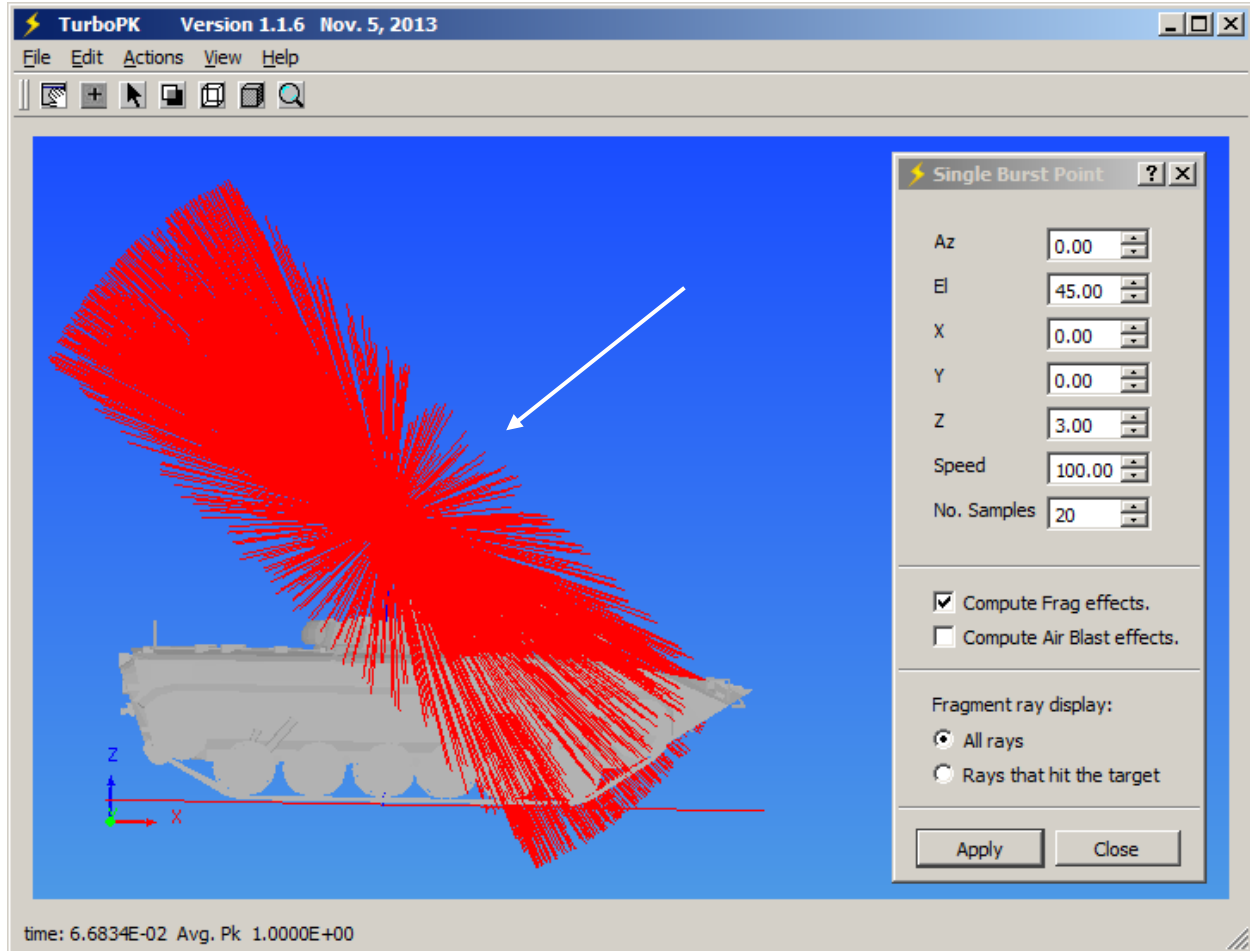


Figure 42 - Fragment rays.

The text in the bottom status bar indicates the total target P_K in this example was 1.0. As in the other options that compute total target P_K , a target "fault tree" file must be loaded, and component $P_{K/H}$ functions must be loaded for this computation to work.

If the ray drawing option is changed to **Rays that hit the target**, and if the menu item **View...View Vulnerable Only** is activated then the image in Figure 43 results. Only rays that have one or more vulnerable components are drawn, and the component individual P_K values (integrated over all impacts) are now used to color code the component's geometry. Only the result of the last Monte Carlo sample will be displayed.

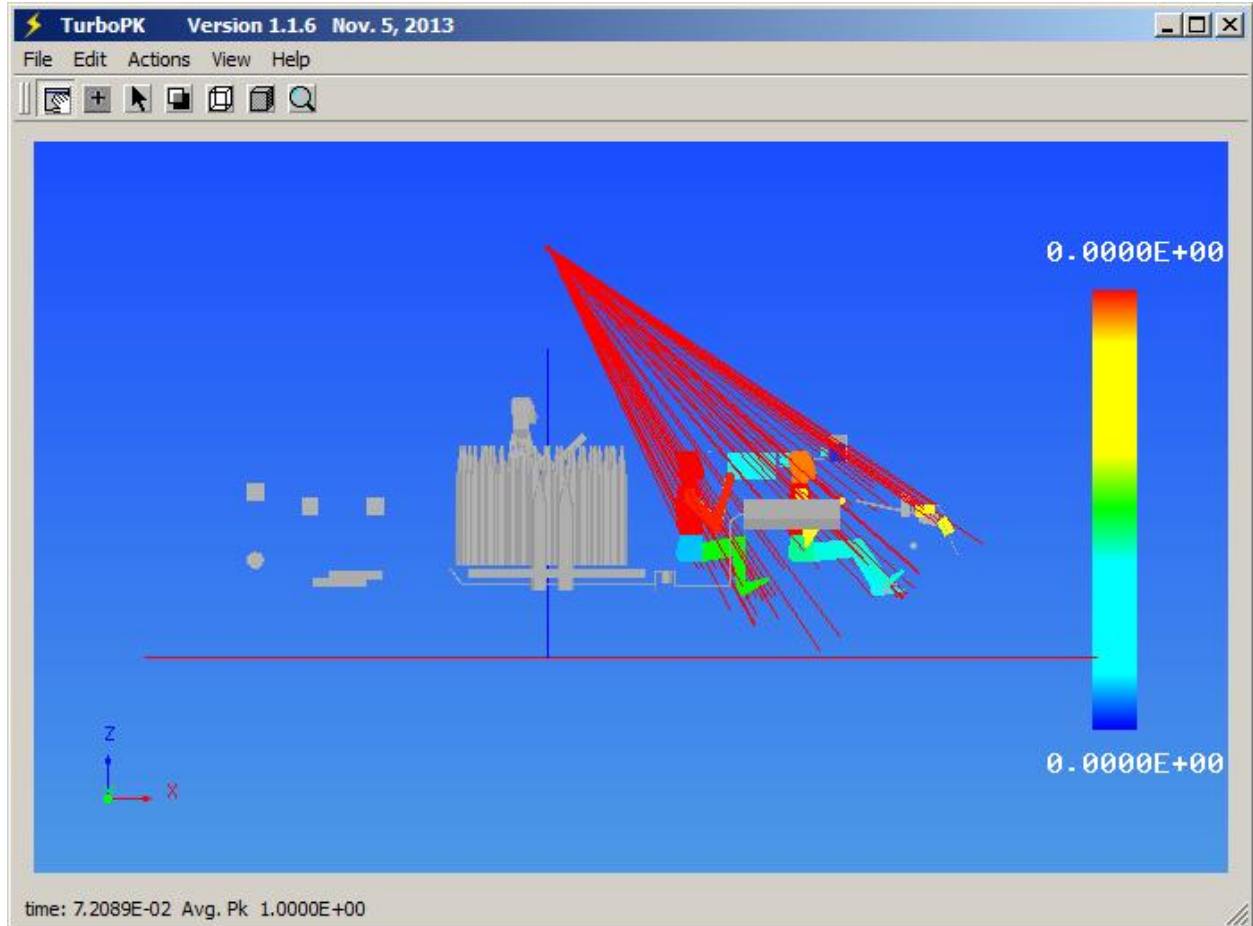


Figure 43 - Displaying component PK values.

8.0 Graphics Display Control

Figure 44 is an image of the graphics toolbar. Most of these capabilities are self explanatory; experiment with them, you'll figure them out.

The selection tool (the arrow icon) works by hovering the cursor over a geometry object and then depressing the left mouse button. The selected object turns red and its ID is noted in the status bar at the bottom of the main window. Clicking on the object a second time de-selects it. The menu item **View...Deselect \ Unhide All** will de-select everything that is in the "selected" state.

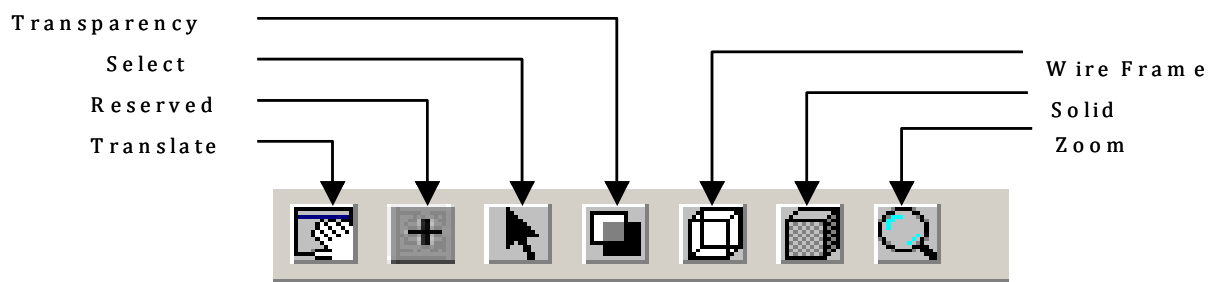


Figure 44 - Graphics toolbar.

In addition the mouse controls rotation of the screen image. Depressing the left mouse button and rolling the cursor rotates the image around the x axis. Depressing the right mouse button and rolling the cursor rotates the image around the z axis. The mouse wheel controls zooming. Once again, experimentation will show how they work.

9.0 Edit Menu

Currently, the **Edit...Point Size** option is the only one that is operative. It allows you to increase / decrease the drawing size of points (in pixels) *after* points have been drawn. The **Edit...Warhead Description** option will be completed in a future release.

9.0 Warhead Files

TurboPK supports two different warhead file formats. The first is the JTCG standard ZDATA format, and the second is a TurboPK-specific format called the "Polar Zone Warhead" format. ZDATA files have the file extension **.zdata**, while Polar Zone Warhead files have the file extension **.pzw**.

ZDATA files are a standard format for fragmentation warhead descriptions. Figure 45 shows the simple example provided with TurboPK displayed in Notepad. The top line in a ZDATA file is a title line and is read in but ignored by TurboPK. The second line *must* contain the string "ZDATA" followed by the number of polar zones in the warhead. In this case there is only one zone. Line 3 contains the polar zone limits (lower, mid, upper), the fragment speeds at the polar zone limits, and the number of fragment mass classes (4 in this case). Line four contains the fragment masses, in grains for the four mass classes. Line 5 specifies how many of each fragment mass class are in the polar zone. Line 6 is the drag coefficient to be used for all fragments in the file. A "real" ZDATA file would have many polar zones but Figure 45 gives you the general idea. ZDATA files contain no information about fragment shape or material type, so TurboPK prompts the user with a dialog box to provide that information when a standard ZDATA file is loaded (Figure 6). Shape factors and shape factor parameters are described in Table 1.

Table 1

Shape Factor	Type	Description
1	Cube	
2	Sphere	
3	Uncontrolled Compact	Modeled as a cylinder with $L/D = 1.0$
4	Parallelepiped	Square cross section. User specifies the edge dimension of the cross-section.
5	Uncontrolled Non-Compact	Modeled as a cylinder with user-specified L/D .

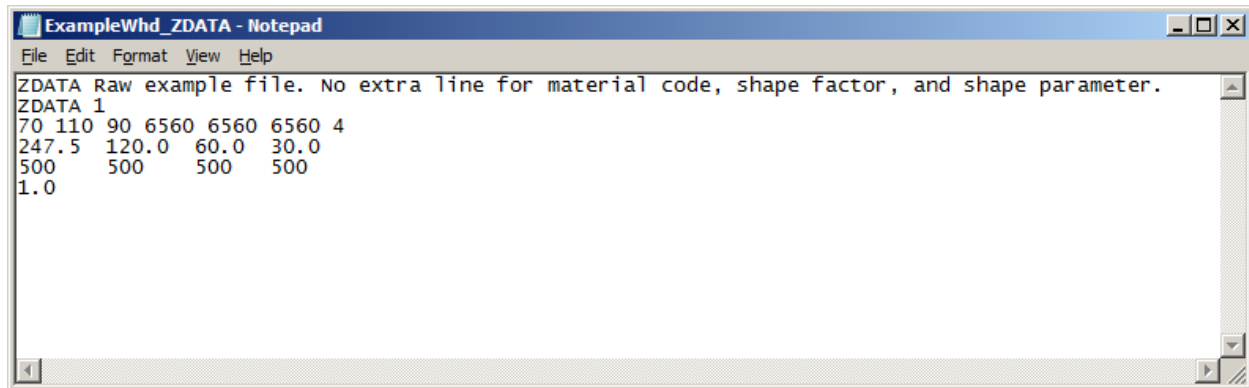


Figure 45 - Example ZDATA file.

For use with TurboPK it is also possible to add material type and shape information as an extra line to a standard ZDATA file (Figure 46). The extra line stipulates the COVART material type for the fragment material (must be 1-6), the shape factor (0-5), and a shape factor parameter for shape factors 4 and 5. The shape factor parameter can be omitted for shapes 1-3. In the case of Figure 33 extra line stipulates material type 4, shape factor 5, and shape factor parameter 3.25.

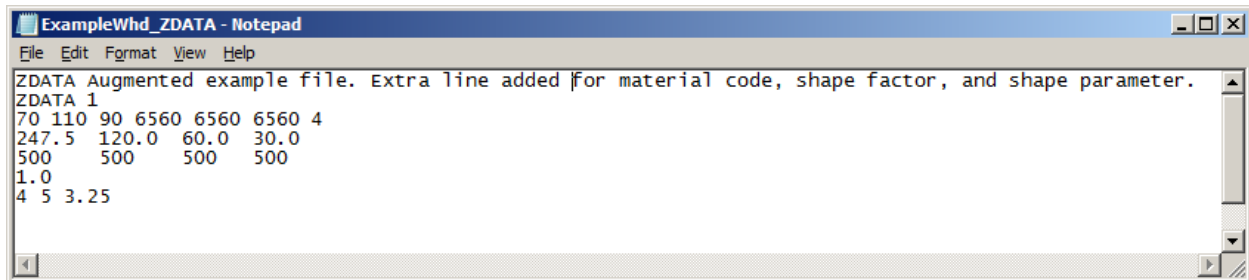
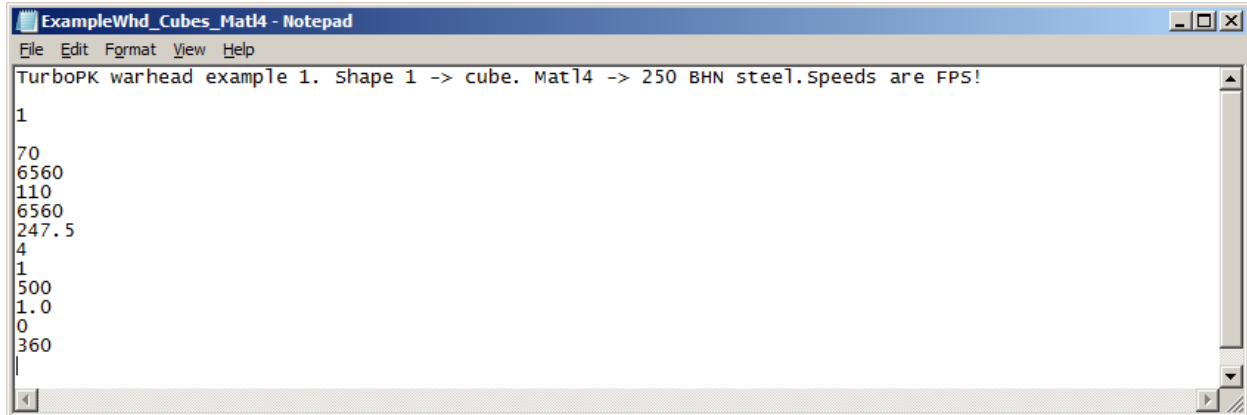


Figure 46 - Augmented ZDATA file.

ZDATA files are limited to one material type and shape for all polar zones and mass classes. The Polar Zone Warhead (PZW) format allows each polar zone and mass class to have its own material code and shape factor. So, for example, the PZW format allows one to mix cubes and spheres in the same file. Figure 47 is an example PZW file. Figure 48 adds commentary at the end of each line (starting with the # symbol).



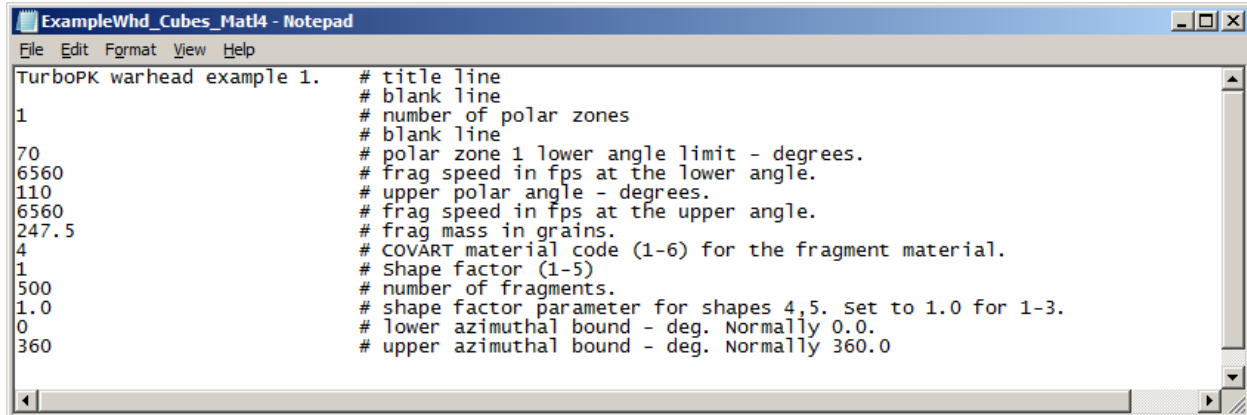
```

ExampleWhd_Cubes_Mat14 - Notepad
File Edit Format View Help
TurboPK warhead example 1. Shape 1 -> cube. Mat14 -> 250 BHN steel. Speeds are FPS!

1
70
6560
110
6560
247.5
4
1
500
1.0
0
360

```

Figure 47 - Example PZW file.



```

ExampleWhd_Cubes_Mat14 - Notepad
File Edit Format View Help
TurboPK warhead example 1. # title line
# blank line
1 # number of polar zones
# blank line
70 # polar zone 1 lower angle limit - degrees.
6560 # frag speed in fps at the lower angle.
110 # upper polar angle - degrees.
6560 # frag speed in fps at the upper angle.
247.5 # frag mass in grains.
4 # COVART material code (1-6) for the fragment material.
1 # Shape factor (1-5)
500 # number of fragments.
1.0 # shape factor parameter for shapes 4,5. Set to 1.0 for 1-3.
0 # lower azimuthal bound - deg. Normally 0.0.
360 # upper azimuthal bound - deg. Normally 360.0

```

Figure 48 - Annotated PZW example file.

The example PZW has only one polar zone, but PZW files can have an unlimited number of zones (limited only by available memory). Each zone is separated in the file by a blank line. Polar zones are completely independent of each other in the file, i.e., they do not have to be in any particular order regarding angle boundaries. Multiple polar zones can have the same angle boundaries, for example, when modeling a polar zone with multiple fragment types.

10.0 Shotline Inspector

This option is for analyzing in detail a single shotline specified by the user. Use the menu item **Actions...Single Ray** to launch the dialog box shown in Figure 49.

Single Ray

Shotline Point

x0

y0

z0

Pick Location

A shotline is defined as a point and a direction vector. Type the point directly or press the "Pick Location" button to use the mouse to designate a point on the geometry. Direction vector is computed from an Az / EL pair. Use the "View Vulnerable Only" menu item to display only vulnerable components.

Direction

dir x

dir y

dir z

Az El

Switch view to this Az El

Fragment Properties

☐ Sphere ☒ Cube

Material

Mass - grains

Speed fps.

Preview Shotline Apply Close

Figure 49 - Single ray dialog box.

The controls grouped on the left side of the dialog control the specification of the shotline. The controls on the right specify a fragment to trace along the shotline.

In this context a shotline is defined as an "aim point" and a direction vector. The aim point is some point through which the shotline is forced to pass, for example a point on the surface of some vulnerable component of interest. There are two ways to define the aim point. The first is to simply type the desired coordinates into the edit controls labeled "x0," "y0," and "z0," respectively. The second is to click on the **Pick Location** button and use the mouse to select the point of interest. Clicking on the **Pick Location** button changes the mouse cursor to a crosshairs shape to indicate that the next mouse click will designate

where the aim point is to be located (Figure 50). A red oval has been added to Figure 38 to highlight the location of the crosshair cursor. Clicking the left mouse button while the crosshair is hovered over the target image causes a ray to be fired "into the screen" as it were, and the first intersection with the target becomes the aim point. This point is displayed as a green point (Figure 51).

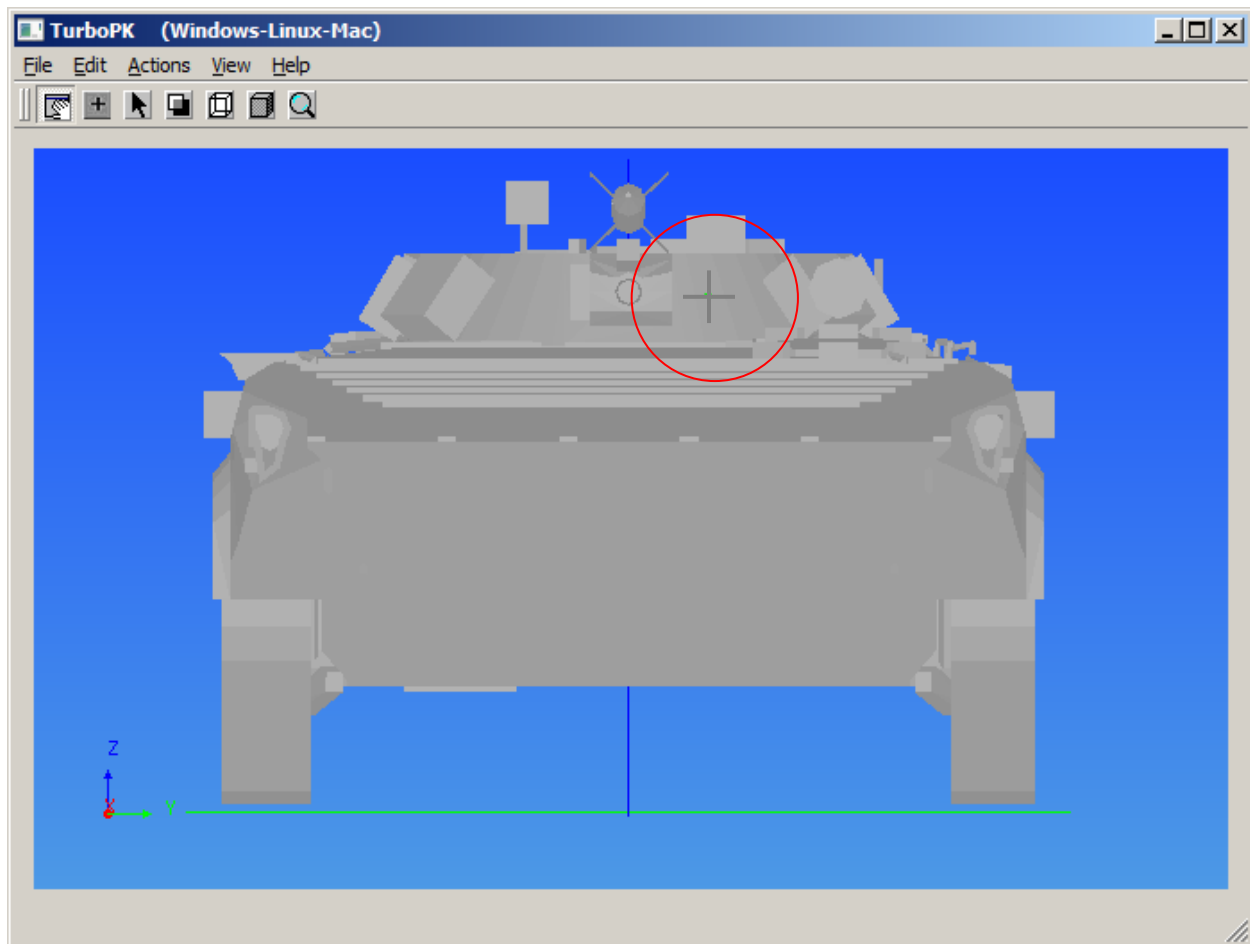


Figure 50 - Selecting An Aim Point.

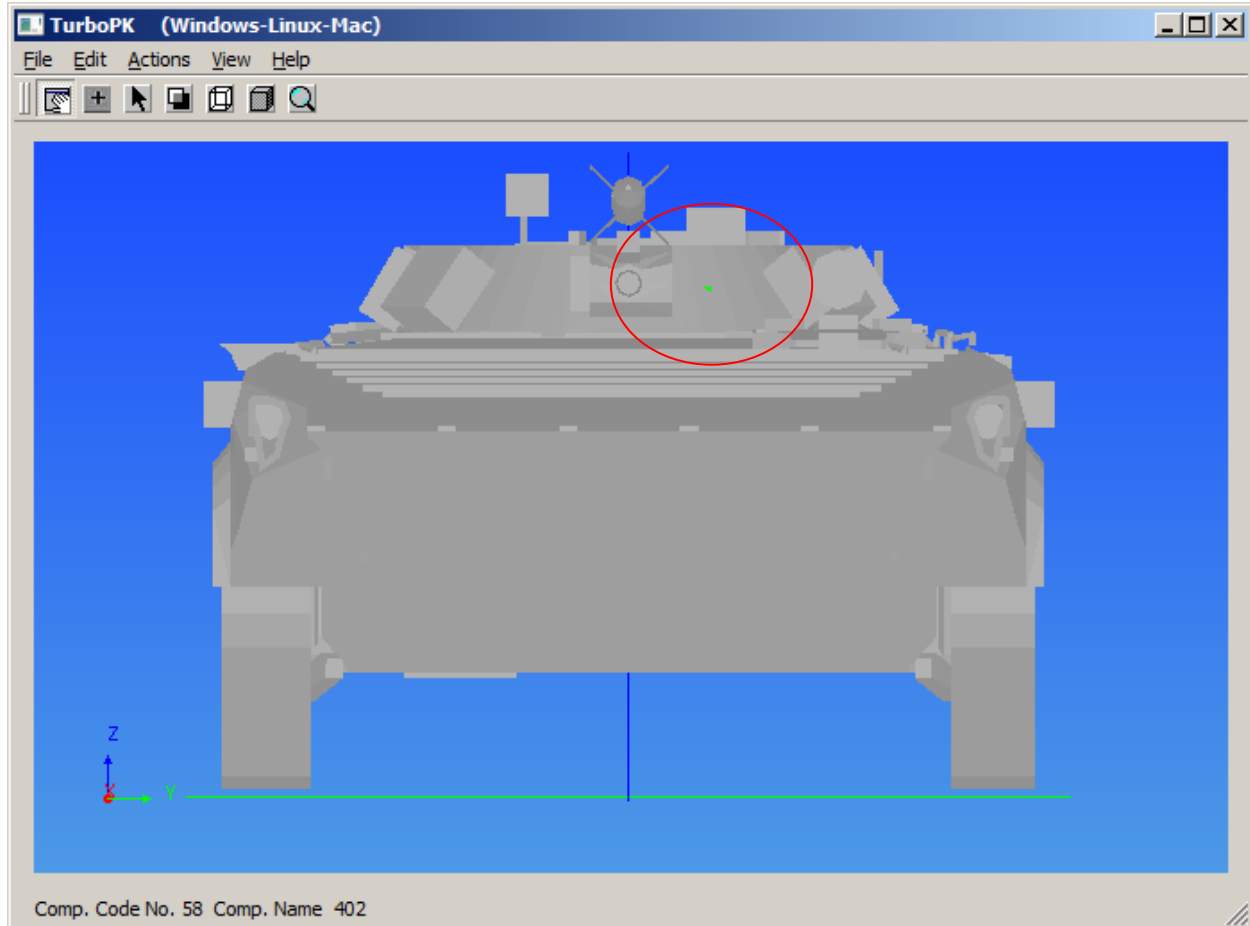


Figure 51 - Aim point marker.

By default the target will be drawn with both azimuth and elevation angles set to 0.0 when the **Actions...Single Ray** menu item is selected. Azimuth and elevation refer to the direction of the shotline in target coordinates. $Az = 0$ and $El = 0$ is "head-on" so the direction vector is $(-1,0,0)$. After picking the aim point shown in Figure 51 the contents of the dialog box are updated to reflect the chosen aim point and direction (Figure 52).

Single Ray

Shotline Point

x0: 0.5764

y0: 0.2777

z0: 1.7752

Pick Location

A shotline is defined as a point and a direction vector. Type the point directly or press the "Pick Location" button to use the mouse to designate a point on the geometry. Direction vector is computed from an Az / El pair. Use the "View Vulnerable Only" menu item to display only vulnerable components.

Direction

dir x: -1.0000

dir y: 0.0000

dir z: 0.0000

Az: 0.0000 El: 0.0000

Switch view to this Az El

Fragment Properties

☐ Sphere ☒ Cube

Material: 4 - Steel BHN 250

Mass - grains: 60.0000

Speed fps.: 5000.0000

Preview Shotline Apply Close

Figure 52 - Aim point coordinates and direction.

Often the component of interest is an internal component, i.e., not visible to the user. To make it visible, first activate the menu item **View...Hide Selected Objects** options then select things to be hidden. Next activate selection mode by clicking on the arrow icon on the toolbar and selecting one or more items. Each time you "select" something it will not be drawn, thus exposing whatever is behind it. (The objects that are not drawn are however still part of the geometry model so they will be used in penetration analysis.) To expose the fragment vulnerable objects use the menu item **View...Fragment Vulnerable**. The effect of this is shown in Figure 53.

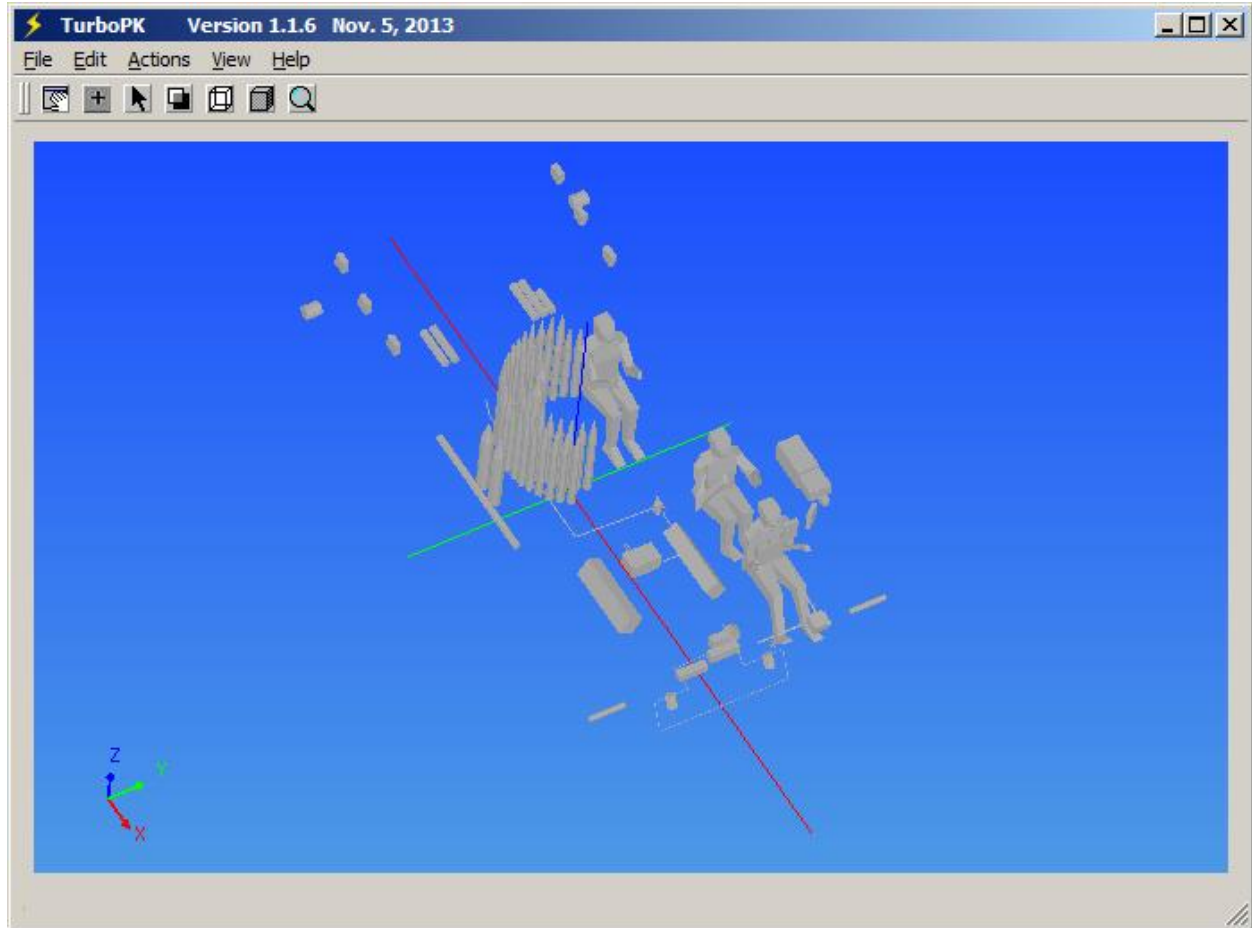


Figure 53 - Fragment-vulnerable components only.

The aim point picking operation only considers objects that are visible thus enabling the user to place the aim point on the surface of internal components.

If the geometry model is rotated around via the mouse operations for rotation, and the user then selects an aim point via the **Pick Location** button, the direction vector and direction angles (azimuth and elevation) will be determined automatically and updated in the edit controls for those variables (Figure 54 and Figure 55).

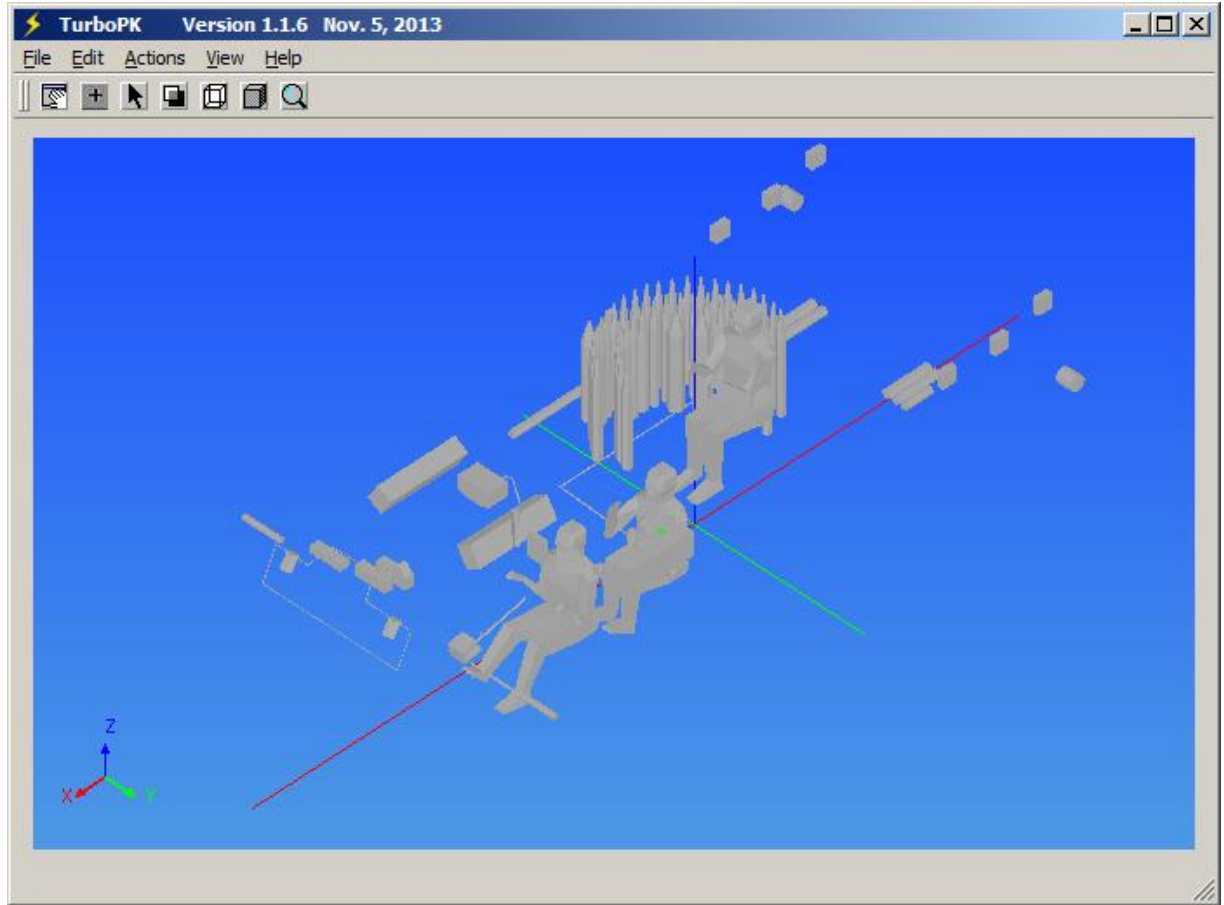


Figure 54 - User rotation defines Az, El, and direction

It is also possible to force a specific Az, El pair by typing the desired values into the Az and El edit controls then clicking on the **Apply (Az, El)** button. If you do that the target will be redrawn at the desired angles and the direction vector controls will be changed to be for the new Az, El values. The direction vector is changed whenever the shotline approach angles are changed. The edit controls **dir x**, etc are read only.

Single Ray

Shotline Point

x0 1.4727

y0 1.1713

z0 1.4986

Pick Location

A shotline is defined as a point and a direction vector. Type the point directly or press the "Pick Location" button to use the mouse to designate a point on the geometry. Direction vector is computed from an Az / EL pair. Use the "View Vulnerable Only" menu item to display only vulnerable components.

Direction

dir x -0.5417

dir y -0.5417

dir z -0.6428

Az 45 El 40

Switch view to this Az El

Fragment Properties

☐ Sphere ☒ Cube

Material 4 - Steel BHN 250

Mass - grains 60.0000

Speed fps. 5000.0000

Preview Shotline Apply Close

Figure 55 - Az, El, and direction vector updated.

To preview the shotline click on the **Preview Shotline** button and a yellow line should appear showing the shotline path through the target (Figure 56). Given an aim point and a direction vector the code will start the shotline outside the target bounds. Because the viewing angle corresponds to looking down the shotline the shotline is not visible until the screen view is rotated away from the shotline angles. That has been done in Figure 56.

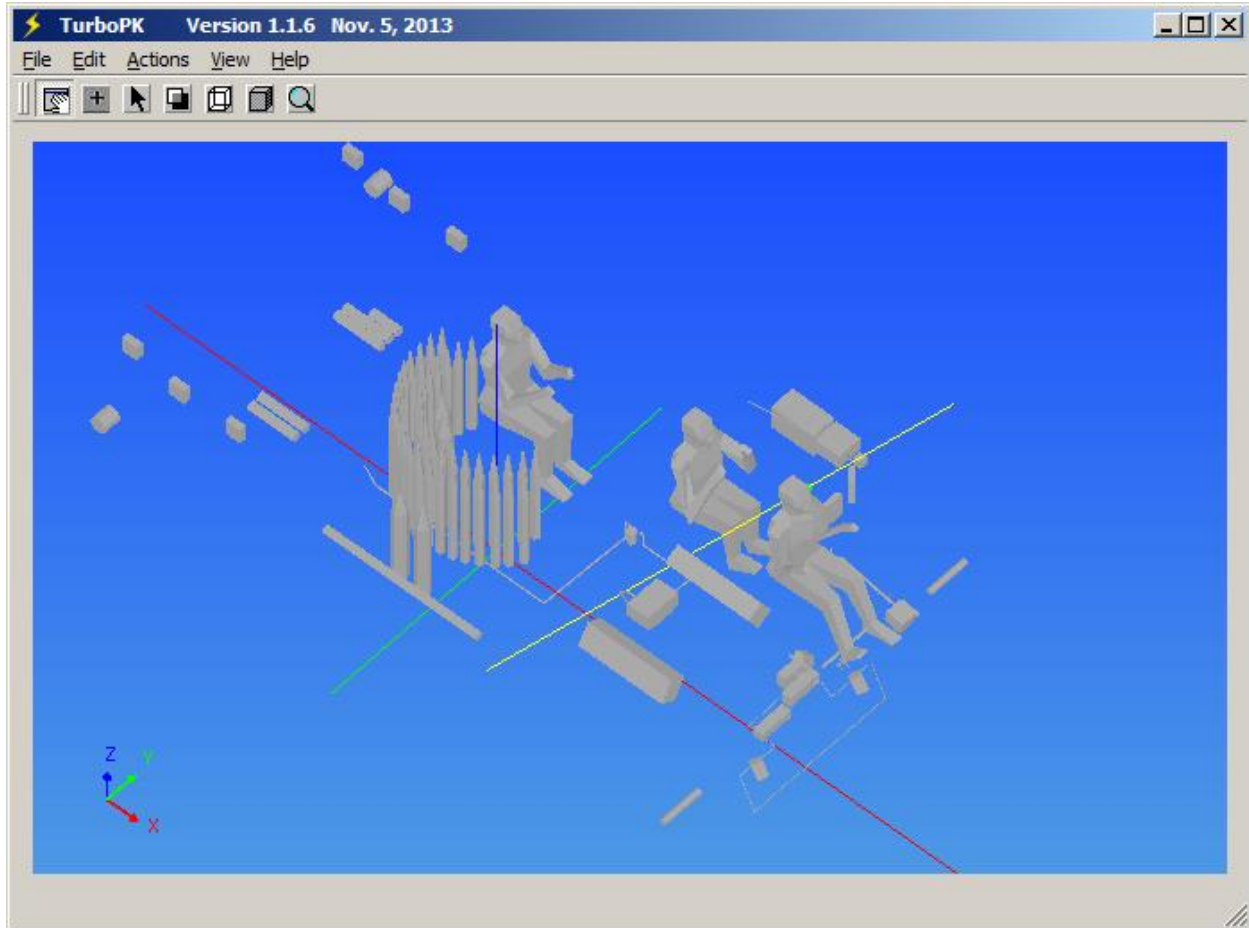


Figure 56 - Shotline Preview

Clicking the **Apply** button causes the code to fire the fragment specified down the shotline specified. At each intersection with a target object the code will record the impact speed, impact mass, exit speed, exit mass, and obliquity. If the component is a vulnerable component it records the $P_{K/H}$ for that impact, and also updates a total shotline P_K . The latter does not invoke the fault tree; it computes a simple P_K by applying the survival rule to all $P_{K/H}$ values encountered to that point. A text window pops up to report the results (Figure 57). The value "Segment Length" is the actual line-of-sight segment length. The value "Modified Length" applies the COVART thickness factor to that if the component is a volume-mode component. Figure 57 is for a 260-grain fragment.

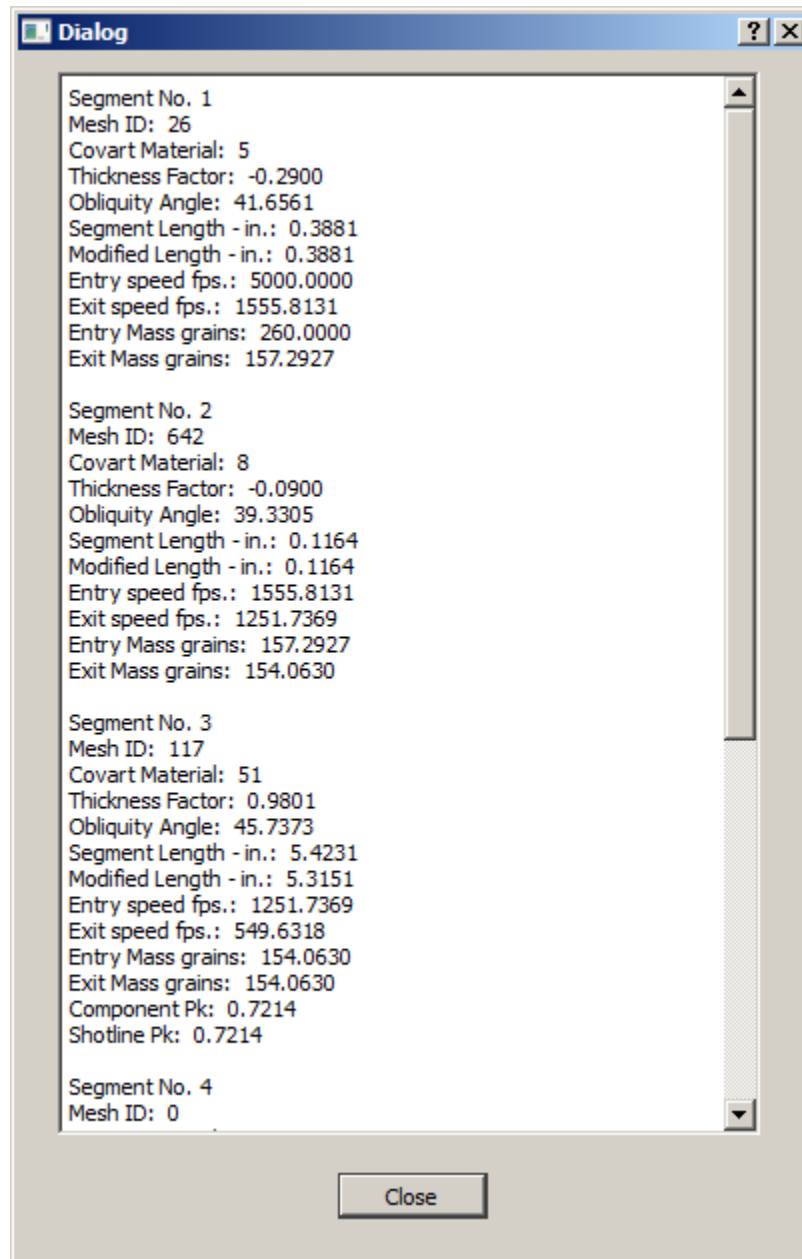


Figure 57 - Shotline history.

11.0 Burst Point Grid Polar Coordinates

Some users prefer to idealize the warhead as being at the origin and the target being displaced w.r.t. the origin. TurboPK now implements this as an option. It is invoked via the menu item **Actions...Burst Point Grid - Polar Coordinates**. Selecting this menu item pops up the dialog box shown in Figure 58. "Weapon Parameters" include projectile orientation and CG location. Currently, only two projectile orientations are offered, (1) nose down, and (2) horizontal. Targets are placed along radial lines emanating from the origin. "Burst Point Grid Parameters" include the angular spacing of the radials, the distance (radius) spacing along each radial, a maximum radius, and the number of Monte Carlo point-bursts to be done at each burst point.

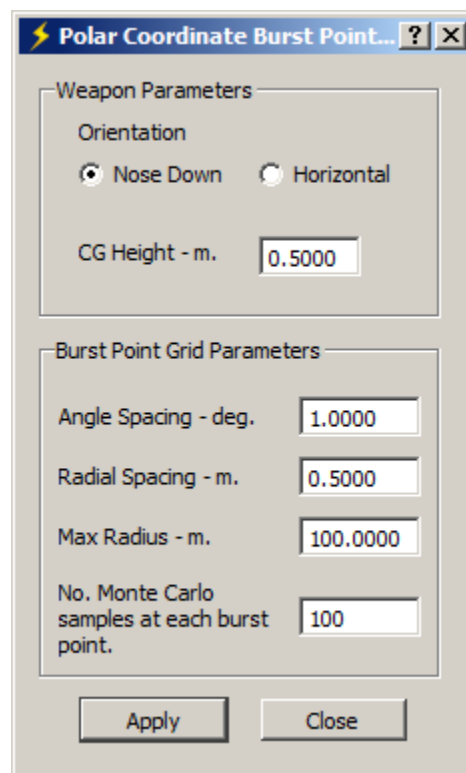


Figure 58 - Polar coordinate option dialog.

Figure 59 shows an example polar coordinate grid calculation. The target in this case was the lightly armored vehicle discussed in previous examples.

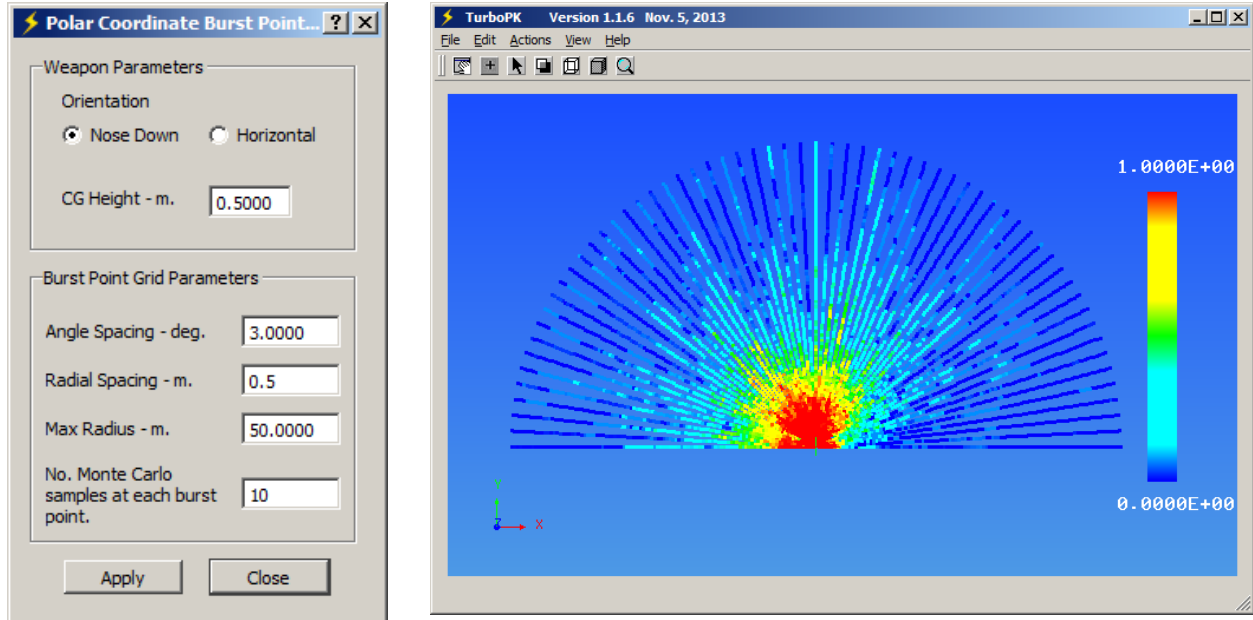


Figure 59 - Example polar coordinate grid calculation.

The current release fixes the weapon elevation angle to either 90-degrees ("Nose Down") or 0-degrees ("Horizontal"). Note that the azimuth angle of approach is not directly specified, so it is by default 0-degrees. A future version of the code will allow the user to specify a particular azimuth angle, or a range of azimuth angles over which the P_K results will be averaged.

12.0 Rectangle of Trajectories Option

This option is invoked from the menu item **Actions...Rectangle of Trajectories** (Figure 60). The associated dialog box is shown in Figure 61. The idea is to generate a set of trajectories, in weapon coordinates, inside a rectangle defined by "dragging a rectangle" via the mouse. (A good example of "Dragging a rectangle" is the selection tool in the Microsoft Paint program.) The general idea for this option is to allow the user to specify a collection of trajectories on or near the target geometry and to examine P_K as a function of position along these trajectories.

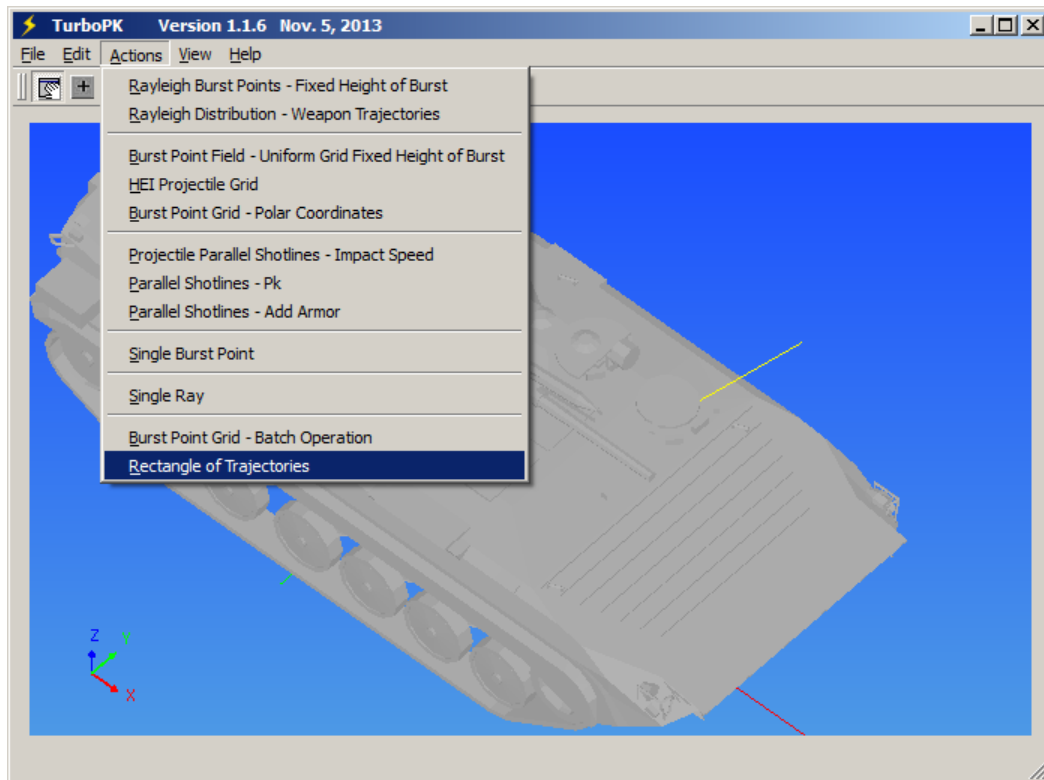


Figure 60 - Invoking the "Rectangle of Trajectories" option.

Rectangle of Trajectories Option	
<div>Define Trajectory Rectangle</div> <div>Generate Trajectories</div>	<p>Click on the Define Trajectory Rectangle button to set the mouse mode to "Drag Rectangle." Then depress the left mouse button and roll the mouse to define the rectangle. Trajectories will be generated "into the screen" as it were.</p> <p>If you modify the Trajectory Spacing value then hit Generate Trajectories to generate the new trajectories.</p> <p>Trajectories are initially computed to start in front of the target and end past the target. Trajectory extents can be extended front and back. Hit Generate Trajectories to apply these extensions.</p>
Trajectory Spacing meters <input type="text" value="0.3280"/> Burst Point Spacing meters <input type="text" value="0.3280"/> Extend Front meters <input type="text" value="0.0000"/> Extend Back meters <input type="text" value="0.0000"/>	
<input type="checkbox"/> Emulate contact fuze. Delay distance - m <input type="text" value="0.0000"/>	<p>"Contact fuze" will terminate trajectories that hit the target at the impact point plus the delay distance. A negative delay distance is a standoff short of the impact point.</p>
No. Monte Carlo samples per burst point <input type="text" value="20"/> Weapon Speed - mps <input type="text" value="0.0000"/> Nose-to-warhead CG distance - m. <input type="text" value="0.0000"/>	<p>Click Analyze Trajectories to compute Pk at the specified Burst Point Spacing along each trajectory.</p> <p>Warhead CG is typically some distance back of the nose contact fuze.</p>
<input checked="" type="checkbox"/> Calculate fragment effects. <input type="checkbox"/> Calculate air blast effects	<p>Select which weapon effects to analyze.</p>
<div>Analyze Trajectories</div>	<div>Close</div>

Figure 61 - Rectangle of Trajectories dialog.

Defining the rectangle starts by clicking on the **Define Trajectory Rectangle** button, which puts the mouse into "drag rectangle" mode. A rectangle is then defined by positioning the cursor in the main window, depressing the left mouse button, and rolling the mouse while keeping the left mouse button depressed. Figure 62 shows an example of this.

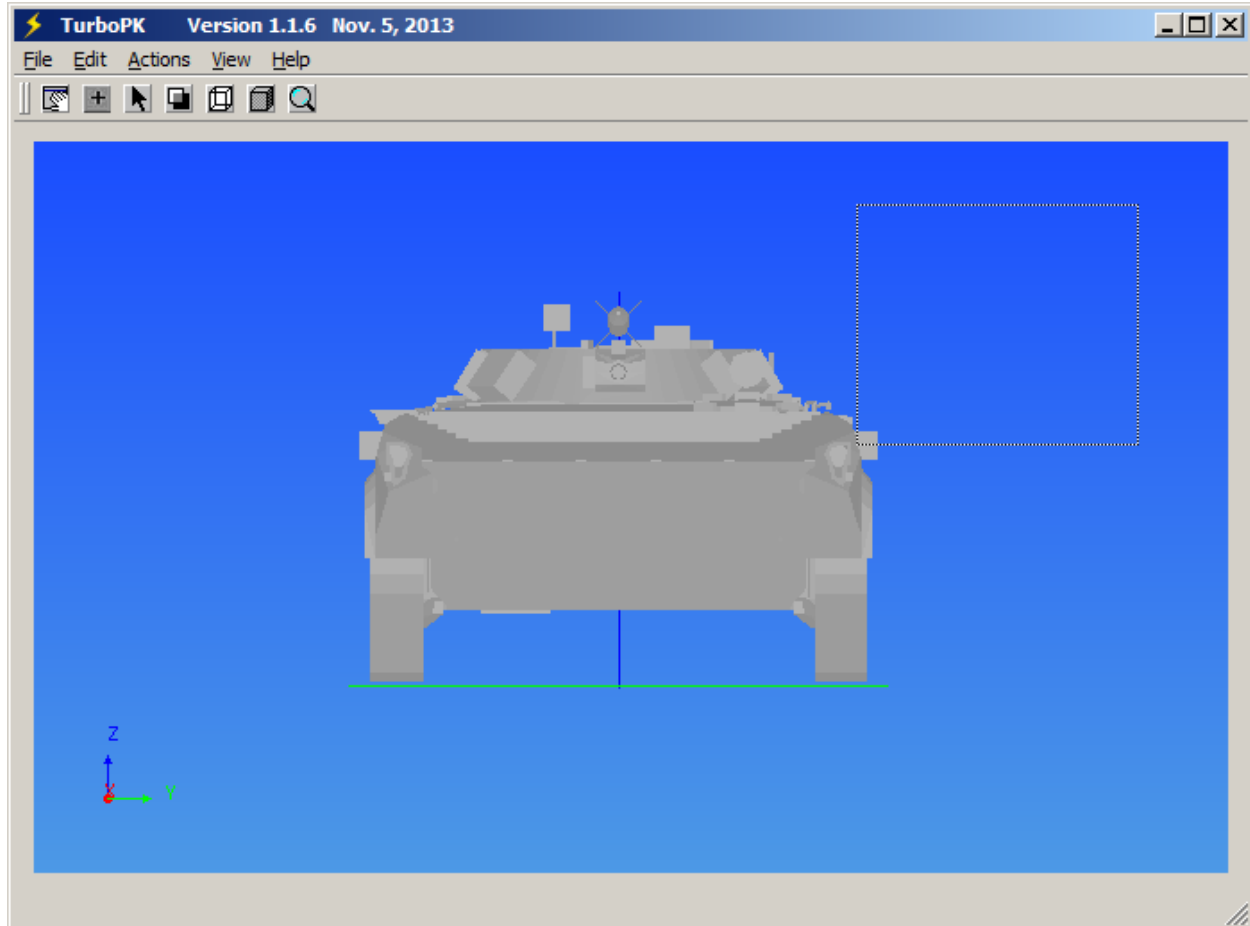


Figure 62 - A rectangle defined on the screen.

In this option the user is in the weapon coordinate system looking at the target. The weapon's path is "into the screen" as it were. The rotational orientation of the target can be set by normal rotation operations via the mouse, or the user can set explicit values for azimuth and elevation angles via the menu item **View...Set Specific Az/El**. Given a rectangle defining the boundaries for a set of trajectories the user specifies several parameters that control the trajectory generation (Figure 60). "Trajectory Spacing" is the spacing of the trajectories within the rectangle. For each trajectory the code will generate a set of burst points along the trajectory spaced at the interval "Burst Point Spacing." The parameters "Extend Front" and "Extend Back" allow the user to lengthen the trajectories.

When the user clicks on the "Generate Trajectories" button a set of lines representing the trajectories is drawn. Rotate the viewing angle to make these lines visible. Figure 63 shows the trajectories generated in the rectangle shown in Figure 62. Note that the trajectories start before they reach the target and end after they pass the target. The cushion is roughly the diagonal distance of the target bounding box. The cushion in front can be increased via the "Extend Front" parameter. The cushion in back can be increased via the "Extend Back" parameter.

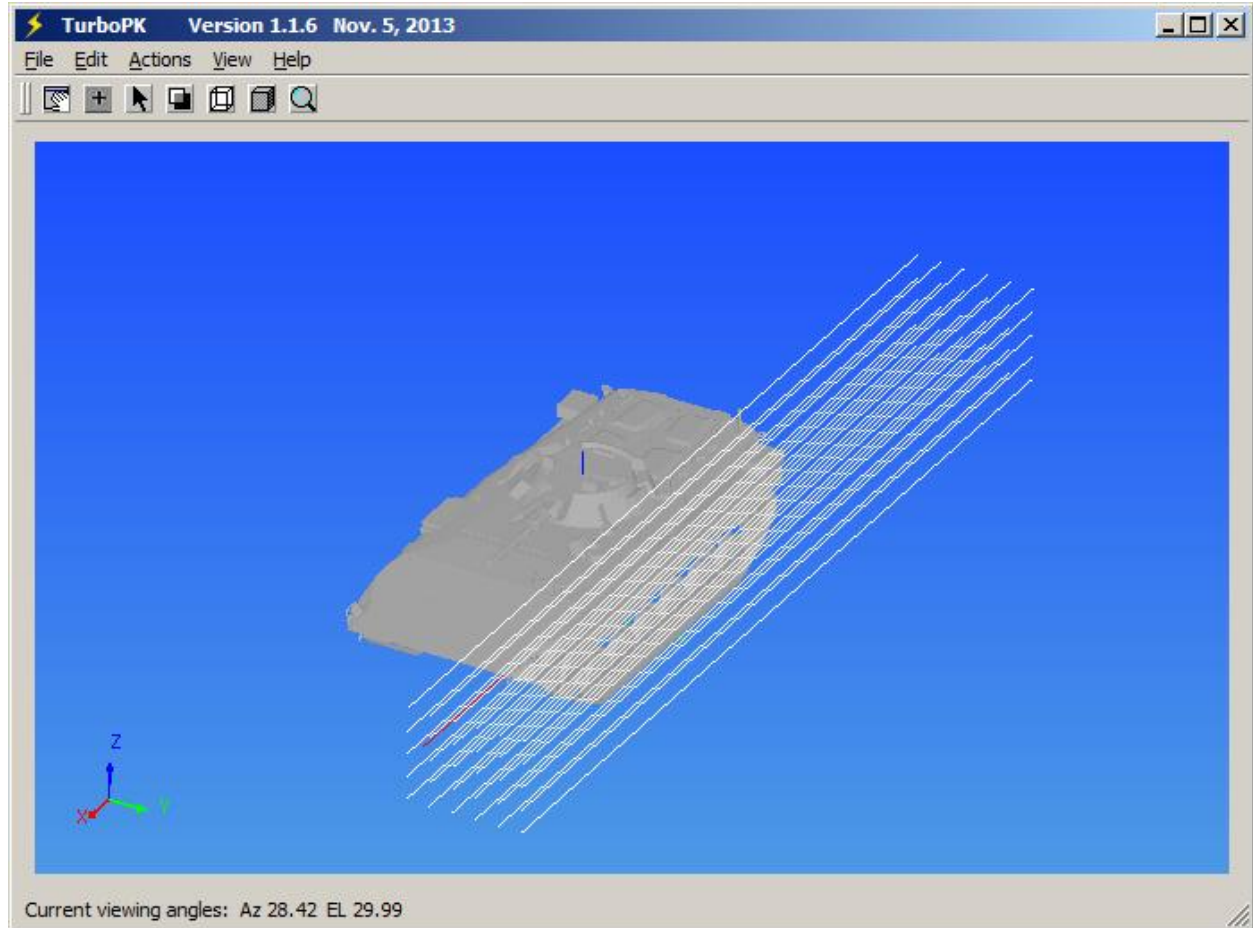


Figure 63 - Rotated view makes trajectories visible.

The other parameters in the dialog box define the number of point-burst samples to be done at each burst location, the weapon's speed, and so forth. These have all been described previously.

Clicking on the button labeled "Analyze Trajectories" causes the burst points to be generated and analyzed. Color coded P_K markers are displayed when the calculations have completed. Figure 64 and Figure 65 show the results for the example rectangle.

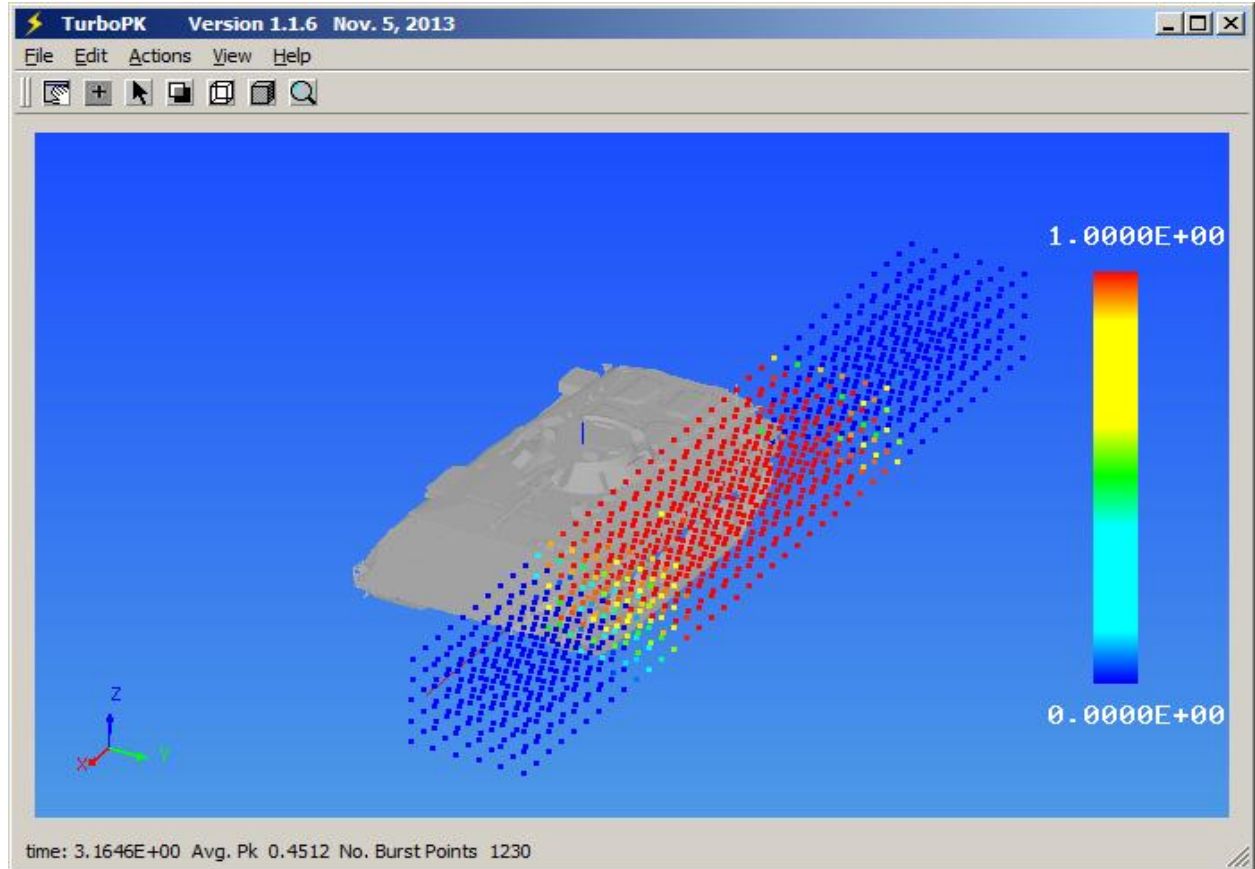


Figure 64 – P_K markers oblique view.

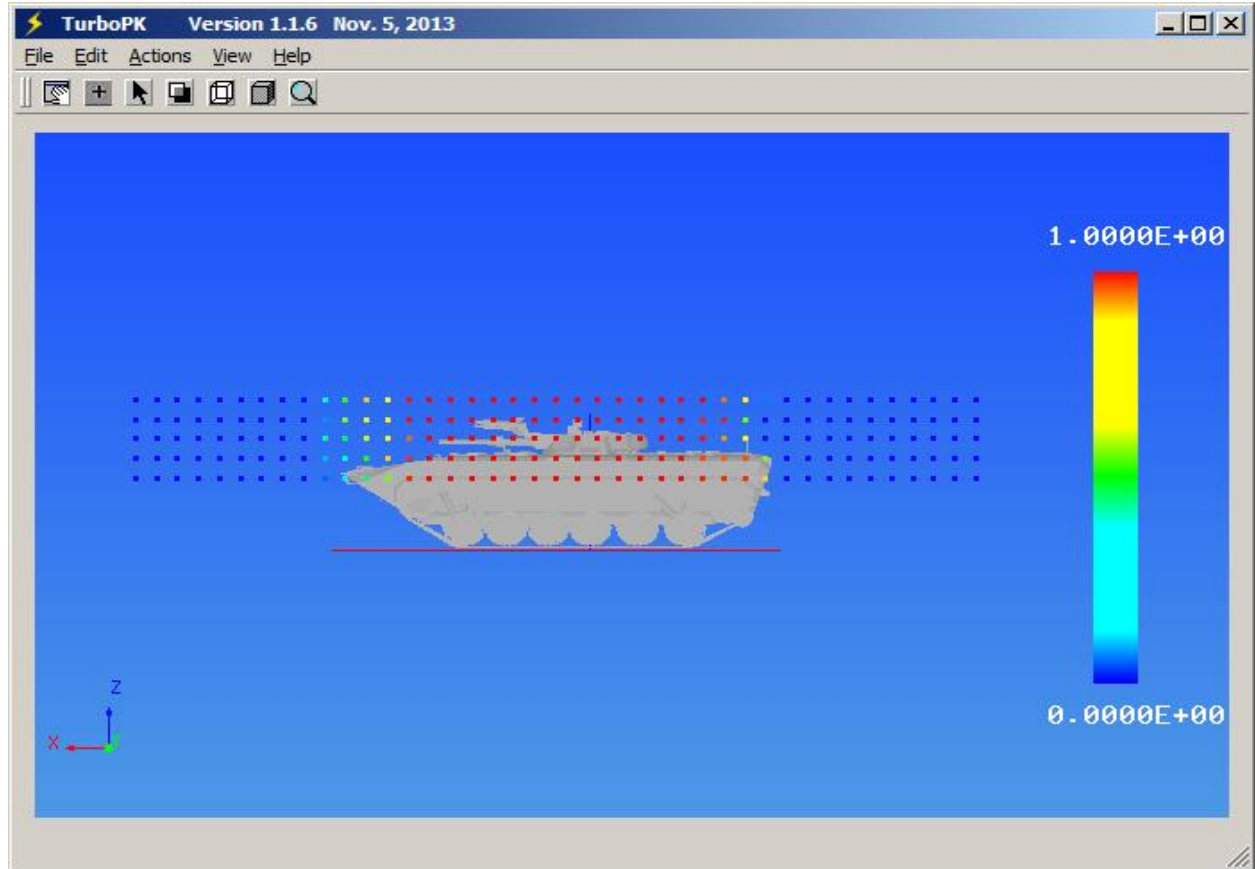


Figure 65 – P_K markers side view.

By default trajectories will pass right through any target geometry they encounter. For example the trajectory rectangle shown in Figure 66 generates the trajectories shown in Figure 67. Figure 67 is restricted to showing just the vulnerable components.

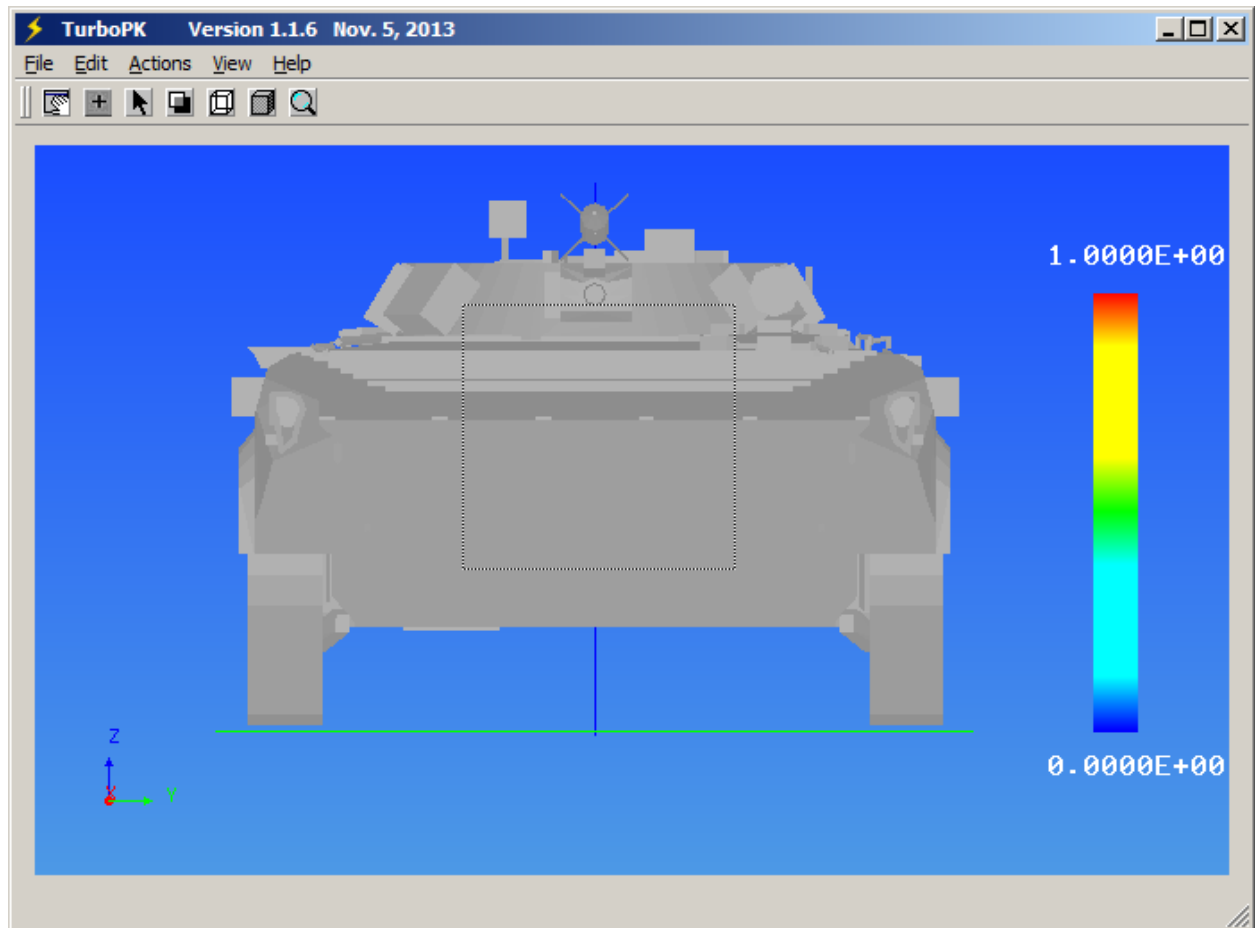


Figure 66 - Trajectories that pass through the target

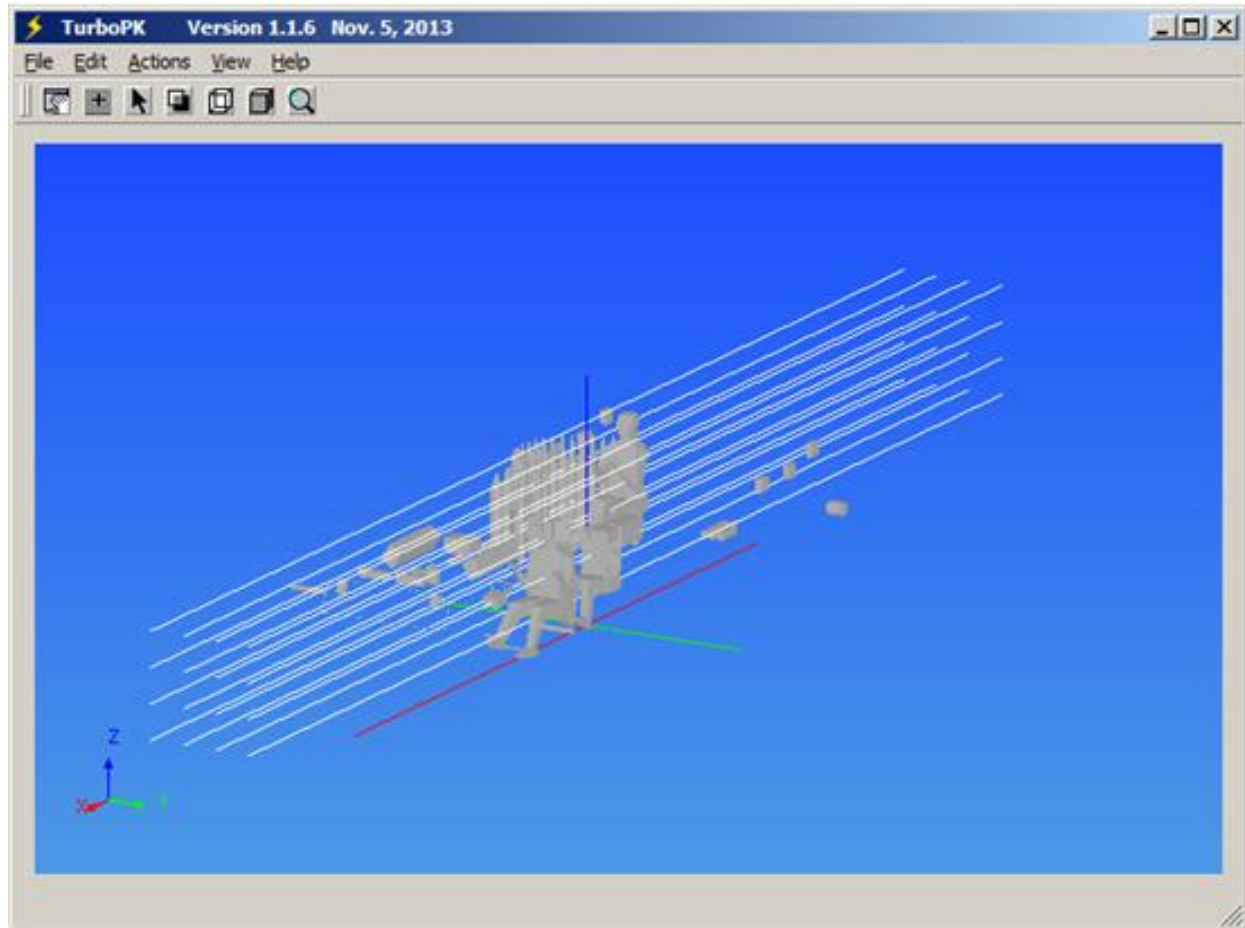


Figure 67 - Trajectories that pass through the target geometry.

Figure 68 shows the P_K results for the example trajectories, and clearly shows burst points internal to the target model.

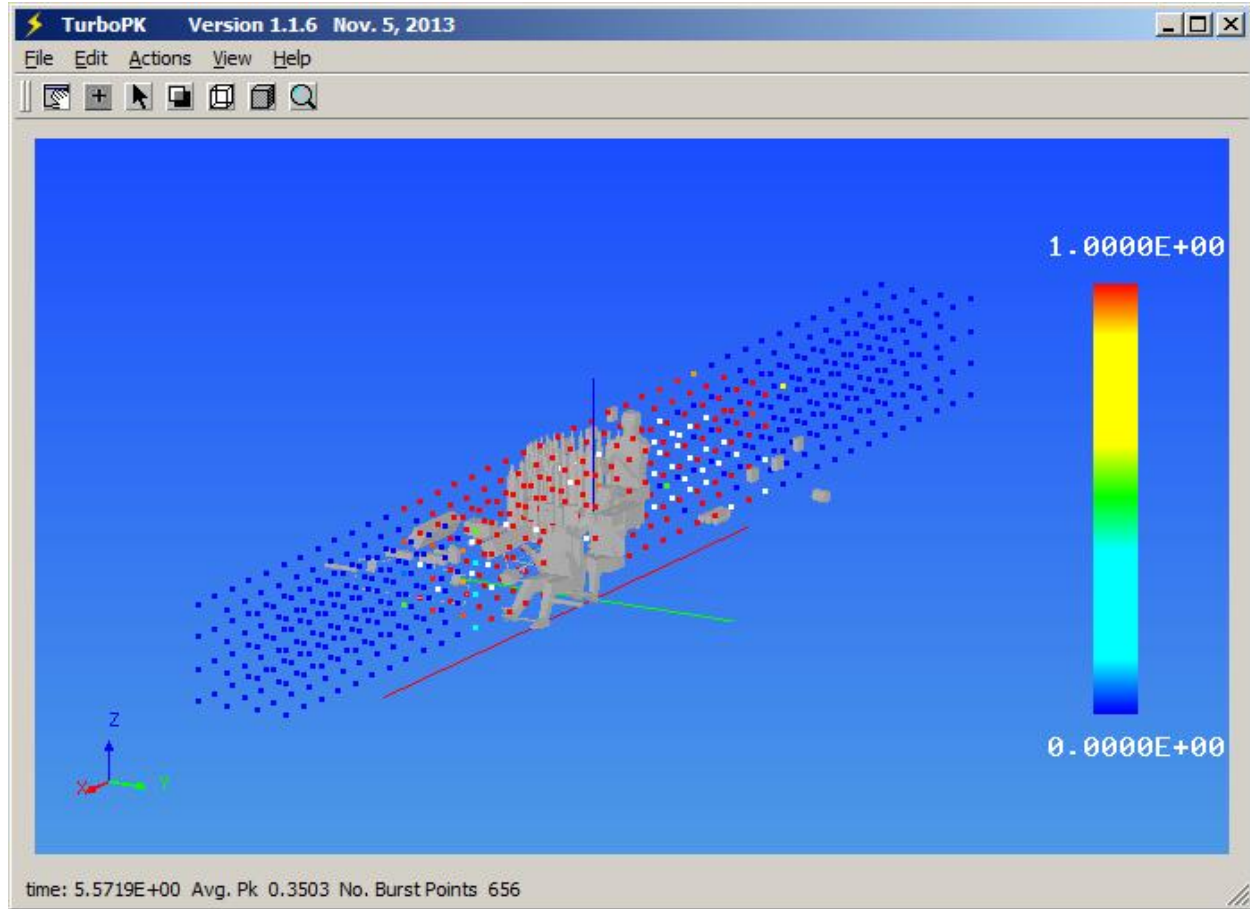


Figure 68 – P_k markers for the example trajectories.

There is an option for emulating a contact fuze. To use it check the box labeled "Emulate Contact Fuze" and specify a delay distance. Under this option trajectories that intersect the target geometry will terminate at the contact point plus the delay distance. Trajectories that do not intersect the target geometry are not affected. Figure 69 and-Figure 70 show an example of emulating a contact fuze with a delay distance of 0.0

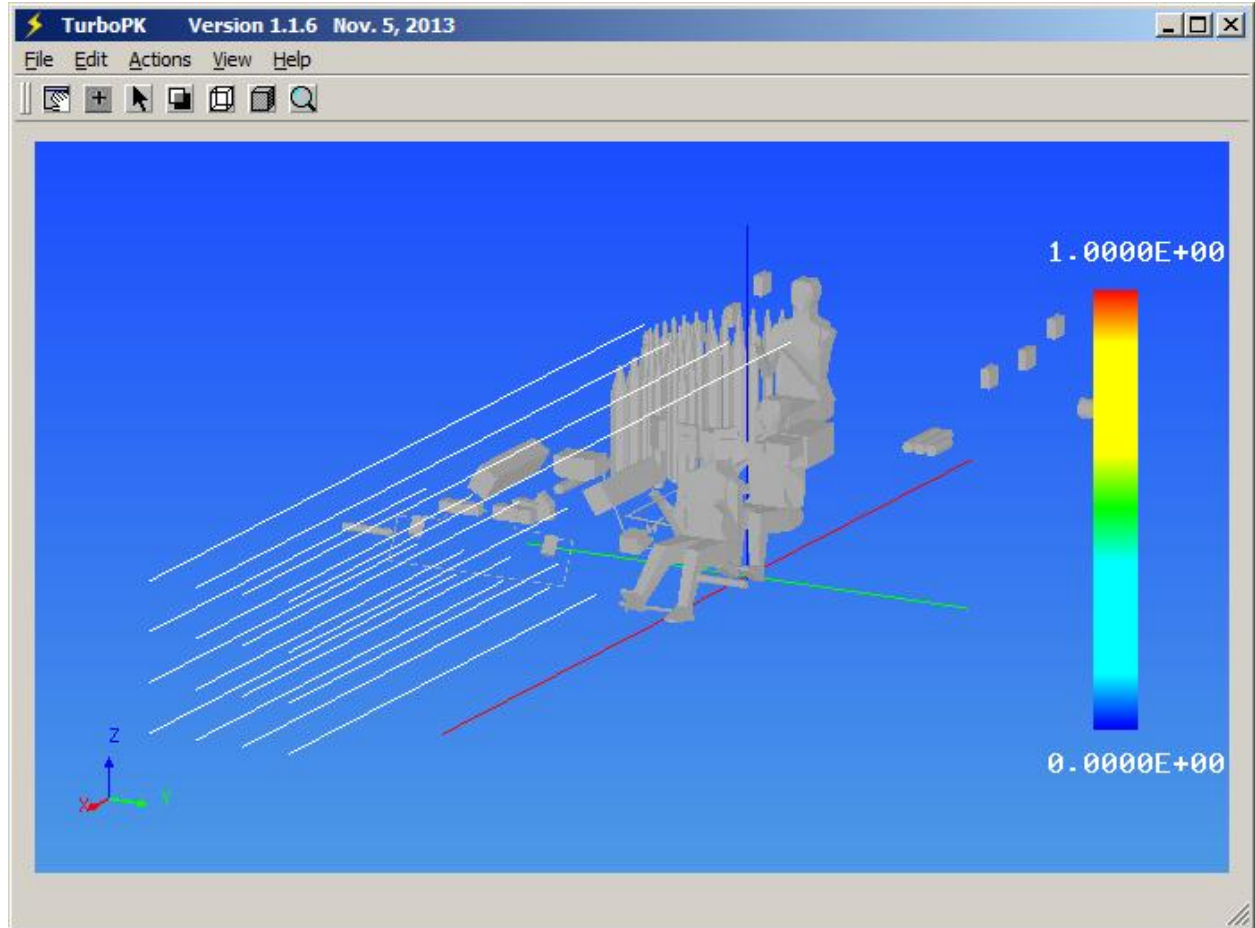


Figure 69 - Trajectories for contact fuze example.

Figure 70 shows the P_K results. In this case the burst points stop where the trajectories first intersect the target. As with the other contact fuze options in TurboPK a positive delay distance moves the burst points past the first point of contact. A negative delay distance moves the burst point back up the trajectory simulating a standoff fuze.

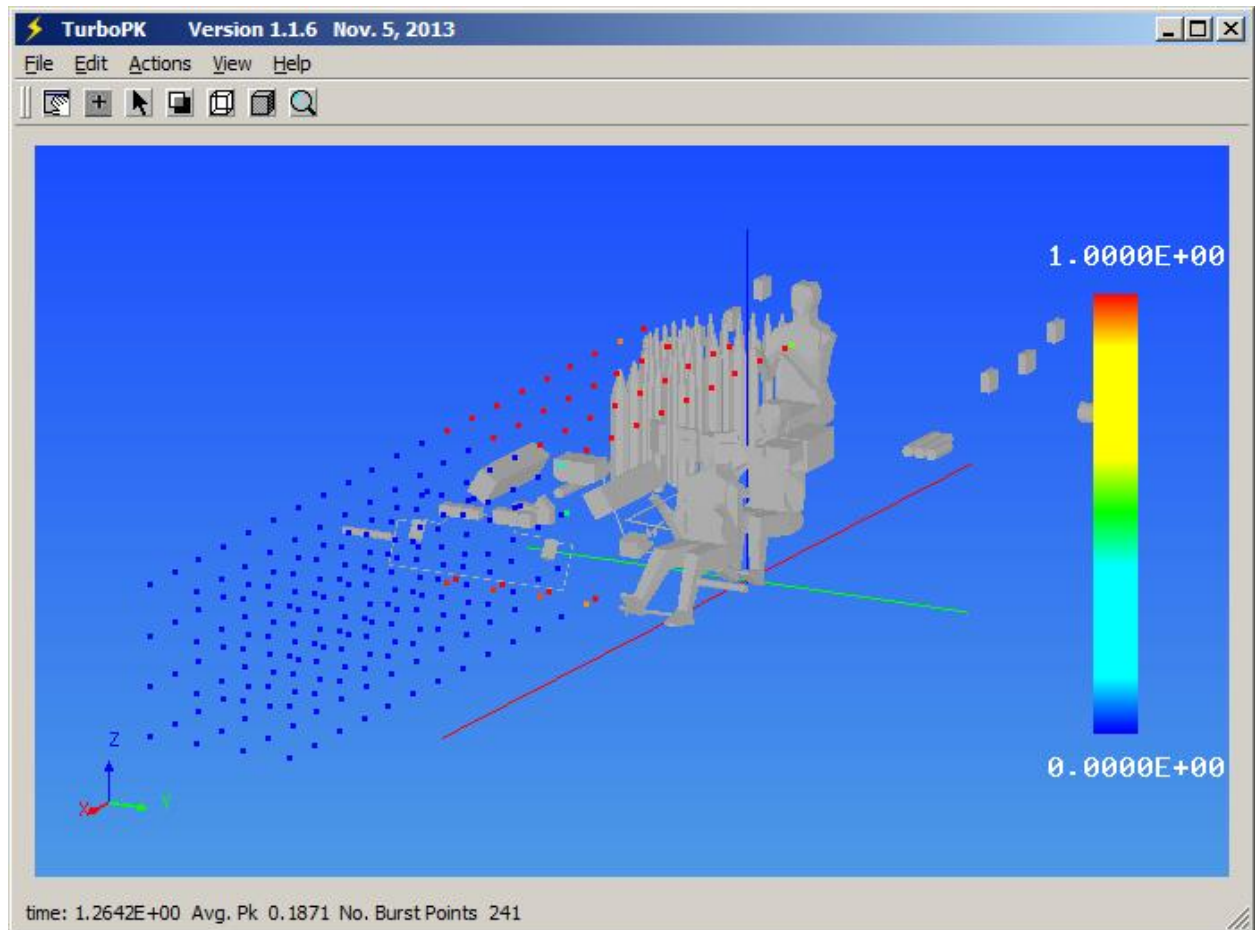


Figure 70 - Contact fuze example.

13.0 Air Blast Model

TurboPK implements the Sadovsky equation for predicting air blast overpressure as a function of explosive mass (TNT equivalent mass) and distance from the charge.

$$\Delta p_1 = 0,84 \frac{\sqrt[3]{m}}{r} + 2,7 \frac{\sqrt[3]{m^2}}{r^2} + 7,0 \frac{m}{r^3}$$

Where:

m = "TNT Equivalent" explosive mass, in Kg.

r = Standoff distance, in meters.

Δp_1 = overpressure, in atmospheres.

The explosive mass is the "energy equivalent" mass of TNT. For example, PBXN-9 has a TNT equivalency factor of 1.6. So 1 Kg of PBXN-9 is equivalent to 1.6 Kg of TNT. Figure 71 shows the Sadovsky equation overpressure for 213 grams of TNT. Pressure is converted to psi in Figure 71. One atmosphere = 14.7 psi. As can be seen in Figure 71, overpressure falls off very rapidly as standoff distances increases.

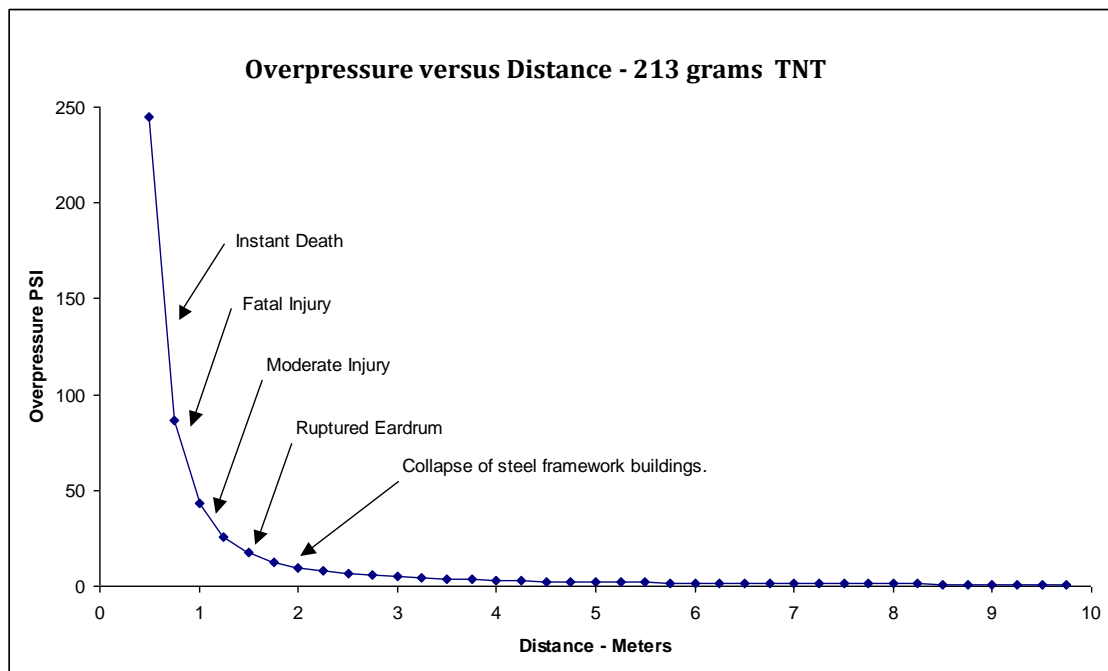


Figure 71

The "TNT Equivalent Weight" parameter is loaded immediately after a warhead file is loaded. The user is prompted by the dialog shown in Figure 72.

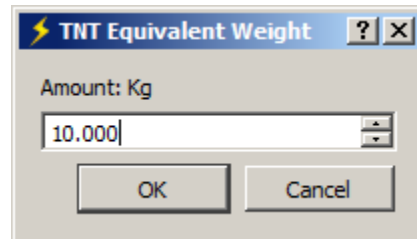


Figure 72 - TNT equivalent weight dialog.

Typically, component damage to air blast is expressed as a critical peak overpressure delivered to the surface of the component. Peak pressure above the critical value results in P_K of 1.0 for the component, and P_K of 0.0 otherwise. This is the model implemented in TurboPK. The user identifies a list of blast-vulnerable components and a critical overpressure for each. Currently this is implemented as an ASCII file that is loaded at run time. Figure 73 shows an example for a handful of components. A complete file would likely be dozens of components.

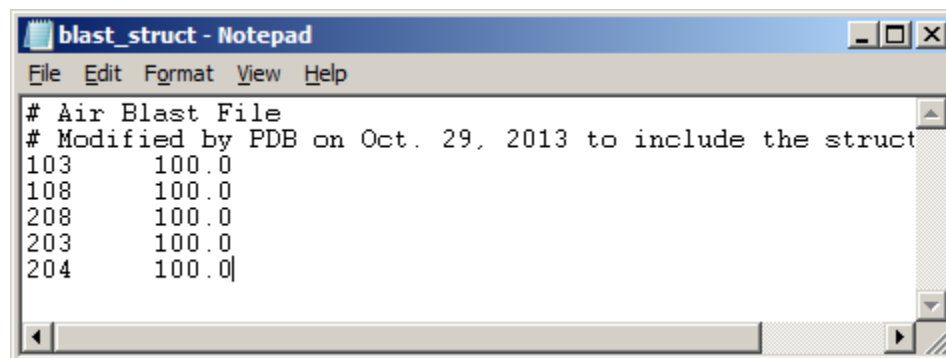


Figure 73 - Example blast vulnerability file.

In this simplified example there are only five blast-vulnerable components. The file format is one component per line containing the object's COVART code number and the critical overpressure required to kill that component. The components do not have to appear in any particular order. A blast-vulnerability file is loaded via the menu item **File...Open Air Blast File** (Figure 74).

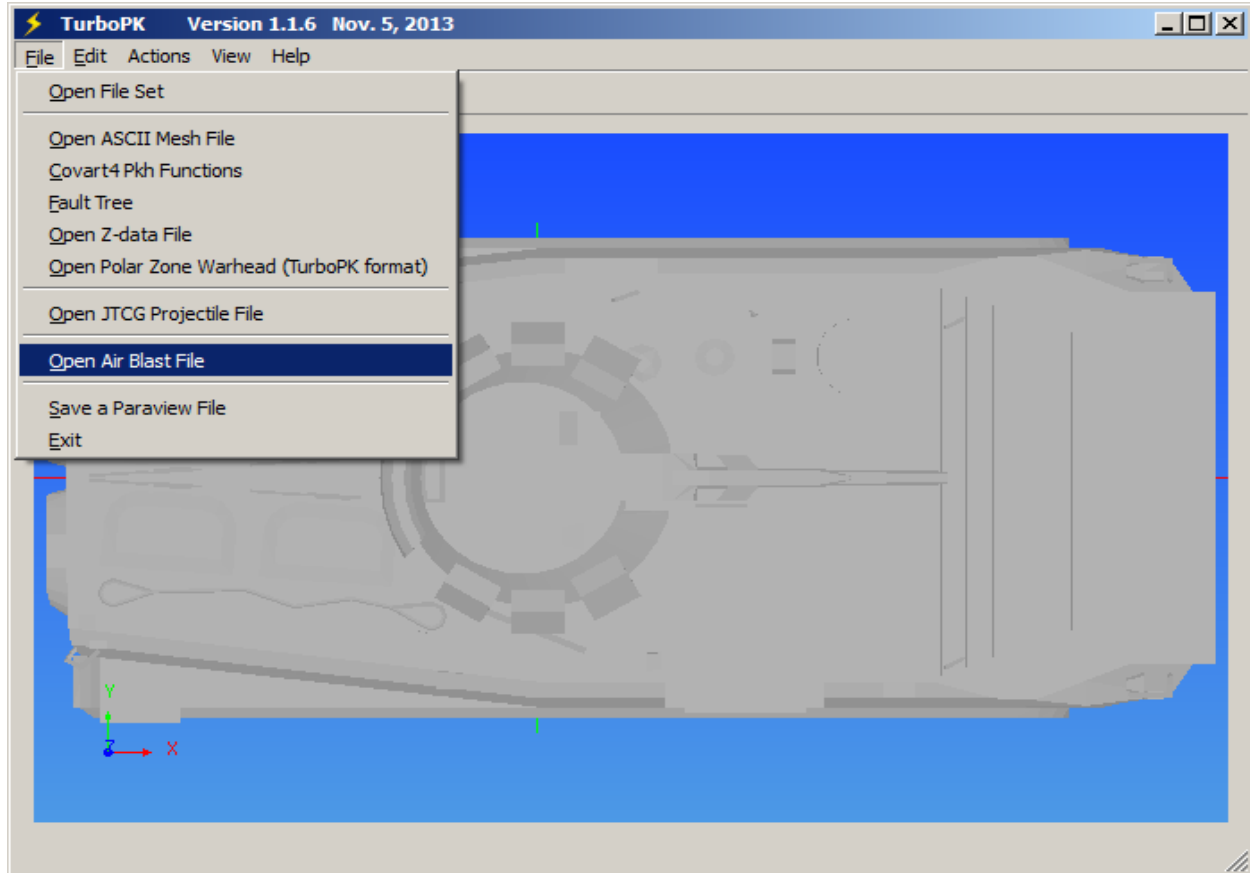


Figure 74 - Loading an air blast vulnerability file.

The **View** menu has been modified to allow the user to display (1) all geometry, (2) all vulnerable geometry, (3) blast-vulnerable geometry, and (4) fragment-vulnerable geometry. Figure 75 shows the blast-vulnerable geometry defined by the example vehicle model. Figure 76 shows the fragment-vulnerable geometry for the same target. Of course, more than the handful of panels shown in Figure 75 are typically blast-vulnerable, and in many cases components are vulnerable to both blast and fragments.

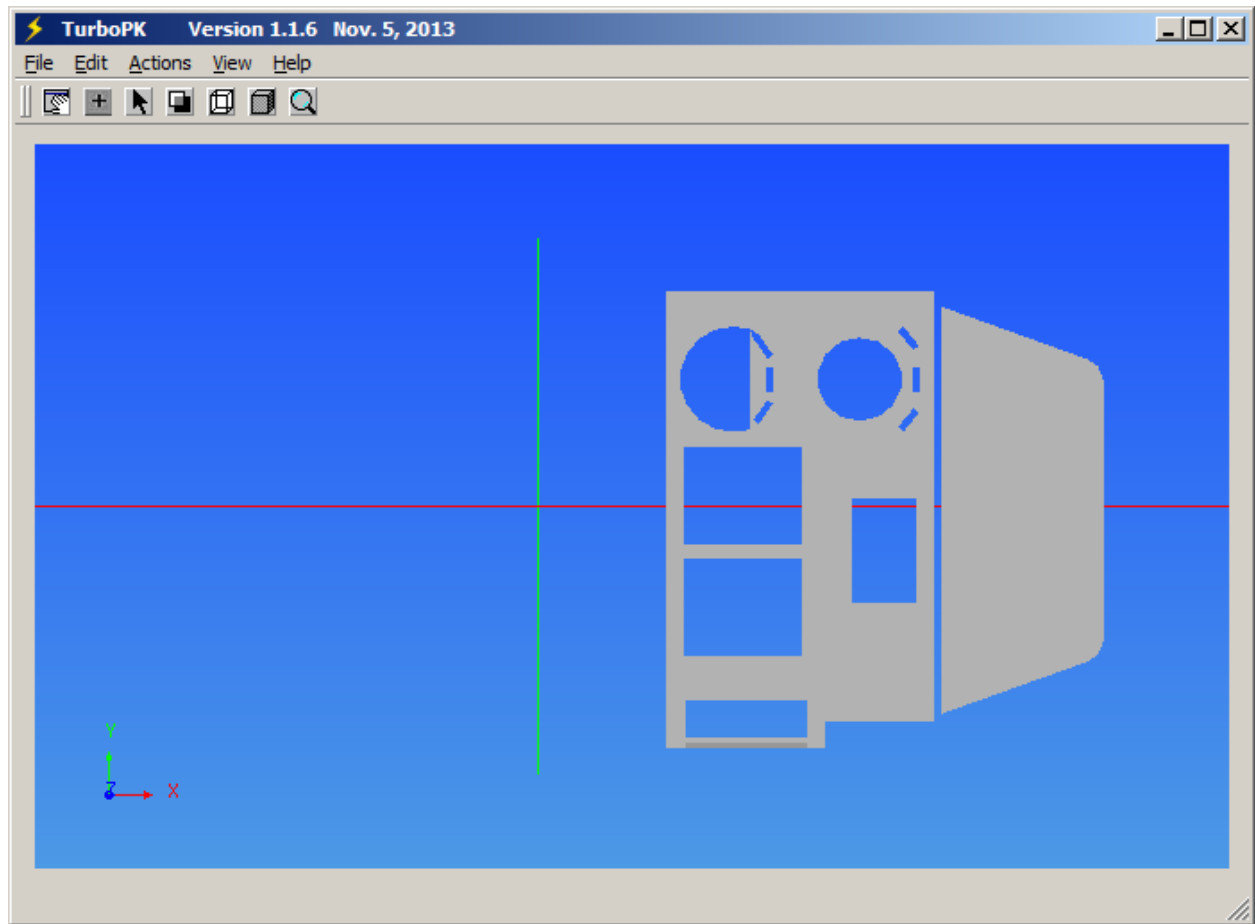


Figure 75 - Blast vulnerable geometry

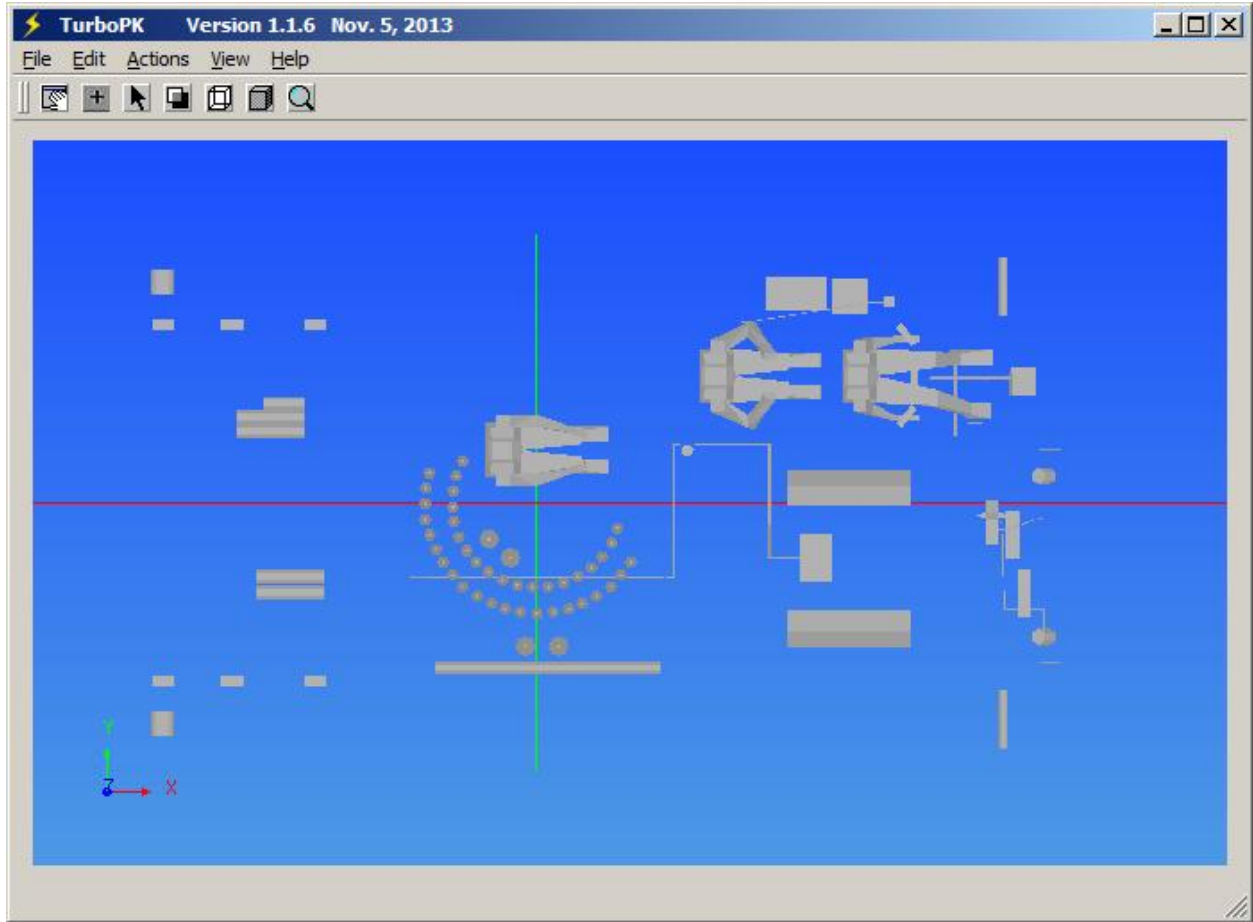


Figure 76 - Fragment vulnerable geometry.

In addition to loading a blast-vulnerability file the user must add the blast-vulnerable components to the fault tree for blast damage to be recognized. If the components in question are already in the fault tree (because they are also fragment-vulnerable) then they do not have to be added again. Only uniquely blast-vulnerable components have to be added. Figure 77 shows a fault tree with a group added to the bottom for blast-vulnerable components (the same ones listed in Figure 73).

```

BMP_CV2_BLAST_TOO - Notepad
File Edit Format View Help

KILL CATEGORY 1
G100=A
SA=1002.OR.1003
END OF GROUP
END OF GROUP
KILL CATEGORY 1
G200=AA.OR.BB.OR.CC.OR.EE
SAA=2001.OR.2002.OR.2003.OR.2004.OR.2005.OR.2006.OR.2007
SBB=2008.OR.2011.OR.2012.OR.2013.OR.2014.OR.2015.OR.2016
SCC=2017.OR.2018.OR.2021.OR.2022.OR.2023.OR.2024
SEE=2025.OR.2026.OR.2027.OR.2028
END OF GROUP
END OF GROUP
KILL CATEGORY 1
G500=A5
SA5=5101.OR.5102.OR.5103.OR.5104.OR.5105.OR.5106.OR.5107\
.OR.5108.OR.5109.OR.5110.OR.5111.OR.5112.OR.5113.OR.5114\
.OR.5115.OR.5116.OR.5117.OR.5118.OR.5119.OR.5120.OR.5121\
.OR.5122.OR.5123.OR.5124.OR.5125.OR.5126.OR.5127.OR.5128\
.OR.5129.OR.5130.OR.5131.OR.5132.OR.5133.OR.5134.OR.5135\
.OR.5136.OR.5137.OR.5138.OR.5139.OR.5140\
.OR.5201.OR.5202.OR.5203.OR.5204.OR.5205.OR.5206.OR.5207\
.OR.5208.OR.5209.OR.5210.OR.5211.OR.5212.OR.5213.OR.5214\
.OR.5215.OR.5216.OR.5217.OR.5218.OR.5219.OR.5220.OR.5221\
.OR.5222.OR.5223.OR.5224.OR.5225.OR.5226.OR.5227.OR.5228\
.OR.5229.OR.5230.OR.5231.OR.5232.OR.5233.OR.5234.OR.5235\
.OR.5236.OR.5237.OR.5238.OR.5239.OR.5240\
.OR.5301.OR.5302.OR.5303.OR.5304.OR.5305.OR.5306.OR.5307\
.OR.5308.OR.5309.OR.5310.OR.5311.OR.5312.OR.5313.OR.5314\
.OR.5315.OR.5316.OR.5317.OR.5318.OR.5319.OR.5320.OR.5321\
.OR.5322.OR.5323.OR.5324.OR.5325.OR.5326.OR.5327.OR.5328\
.OR.5329.OR.5330.OR.5331.OR.5332.OR.5333.OR.5334.OR.5335\
.OR.5336.OR.5337.OR.5338.OR.5339.OR.5340.OR.5501.OR.5502\
.OR.5503.OR.5504.OR.5511.OR.5512.OR.5513.OR.5514\
.OR.5621.OR.5622.OR.5623.OR.5624\
.OR.5701.OR.5702.OR.5703\
.OR.5704.OR.5705.OR.5751
END OF GROUP
END OF GROUP
KILL CATEGORY 1
G700=A7
SA7=7002.OR.7003.OR.7101.OR.7104.OR.7105
END OF GROUP
END OF GROUP
KILL CATEGORY 1
G900=BLAST
BLAST=0103.OR.0108.OR.0203.OR.0204.OR.0208
END OF GROUP

```

Figure 77 - Blast-vulnerable components in a fault tree.

The **Edit** menu now has a "Write a default air blast file," which is a helper function for creating an air blast file. When this menu item is invoked it asks the user to specify an output file name then pops up the dialog box shown in Figure 79.

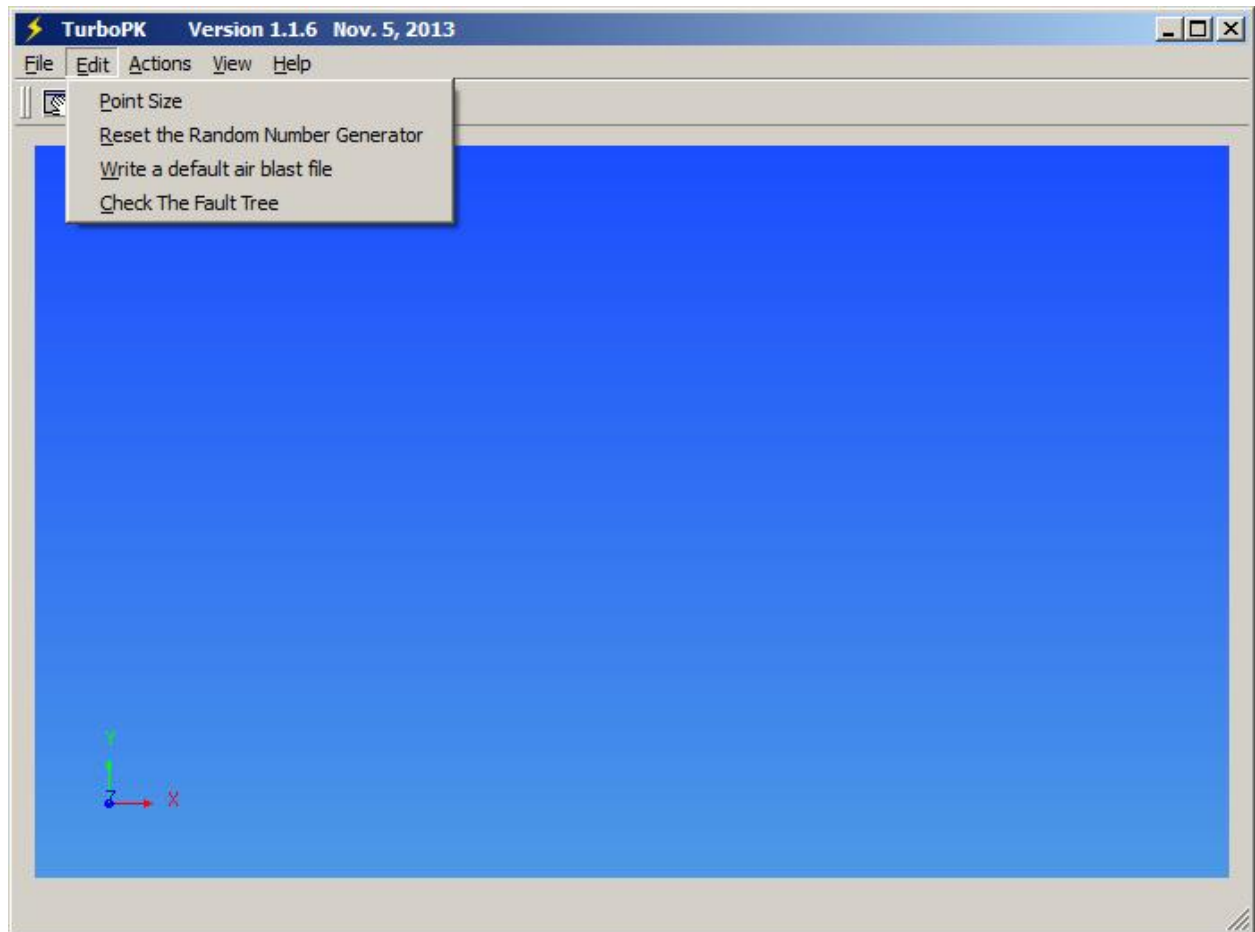


Figure 78 - The Edit menu.

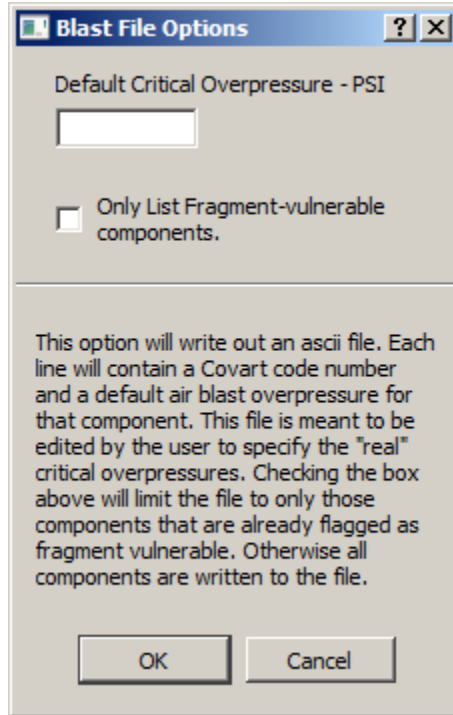


Figure 79 - Default air blast parameters.

This dialog allows the user to define a default critical overpressure for blast vulnerable components, and also allows the user to restrict the components written out to be those that are already vulnerable to fragments. (Note: Most structural elements are typically *not* considered vulnerable to fragments so leave this box unchecked if you want the structural elements written out to the file too.)

14.0 Miscellaneous

The **Edit** menu shown in Figure 78 has an item labeled "Point Size" which pops up a dialog box for defining the size, in pixels, of points that are drawn on the screen (for example, P_K markers). The menu item labeled "Reset the Random Number Generator" does exactly that. The menu item labeled "Check the Fault Tree" will scan the fault tree for items that are not flagged as vulnerable to either fragments or air blast. While technically not an error, including non-vulnerable components is probably not what the user intended. If any are found a text dialog will pop up listing the names of the suspect items in the fault tree.

The **View** menu now contains an item labeled "Set Specific Az / El" which pops up a dialog box for specifying the Az / El pair (Figure 80). In this case the Az / El being specified are the *viewing* angles, not weapon approach angles.

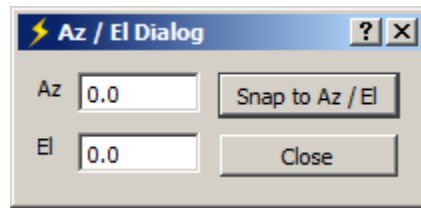


Figure 80 - Setting specific viewing angles.

The **View** menu also has an item labeled "View List of Files Loaded" which will show the user which files have been loaded for the current computing session. For example, Figure 81 shows the list of files loaded from the example **Vehicle** folder.

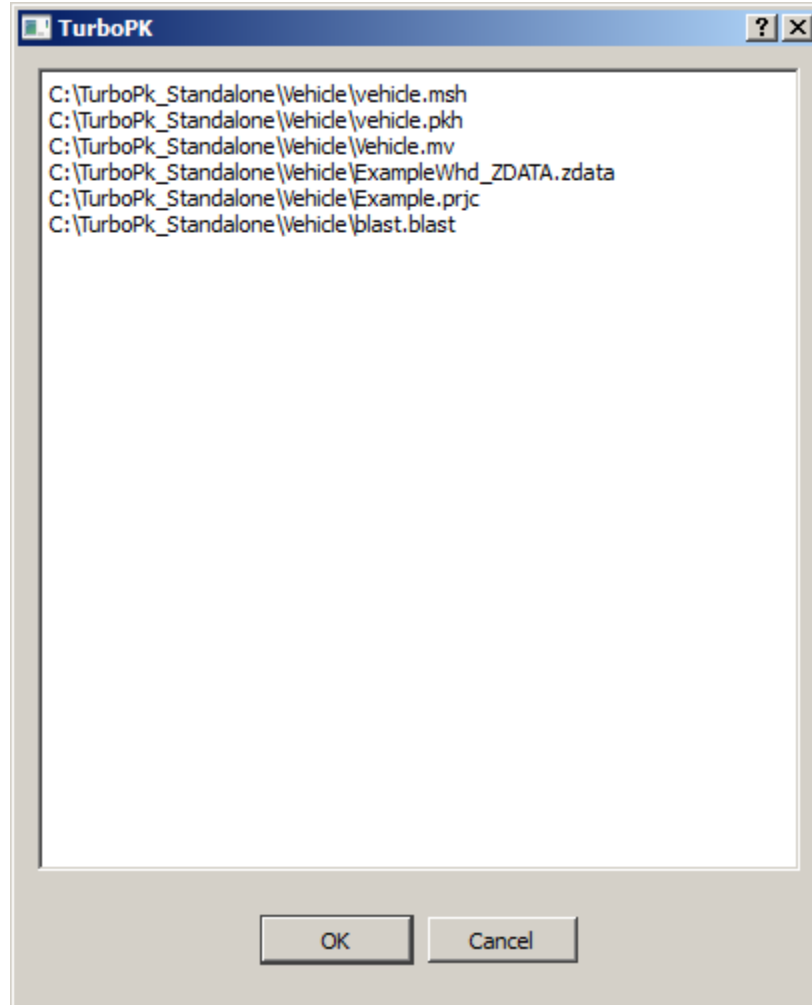


Figure 81 - Listing files that are currently in use.