g4.prim format version 2.4
for Fukui Renderer DAWN

Satoshi Tanaka

February 5, 1998

1 Introduction

This document explains the "g4.prim format" which is able to describe visualizable 3D (three-dimensional) scenes transferred from GEANT4 with help of its visualization manager and Fukui Renderer driver (DAWN driver). In the following, we use the term "3D data" meaning a data to construct a 3D scene or a hint information for the construction, e.g., shape, color, body-coordinate definition, and bounding box of the whole scene. On the other hand, the term "g4.prim-format data" means a set of 3D data plus several rendering commands to control DAWN.

DAWN receives the g4.prim-format data from GEANT4 via the TCP/IP socket of the default port number 40701, or via the named pipe by specifying -G and -g options, respectively:

```
% dawn -G     (invoking DAWN with socket mode)
% dawn -g     (invoking DAWN with named pipe mode)
```

GEANT4 invokes DAWN with -G option by default, and with -g option if the environmental variable "G4DAWN_NAMED_PIPE" is set to 1. How GEANT4 sends the g4.prim-format data to DAWN is demonstrated in the test programs, "`g4test_inet.cc`" and "`g4test_unix.cc`", included in the DAWN package. G4.prim-format data sent from GEANT4 are automatically saved to a file with the name "g4.prim" in current directory of DAWN.

G4.prim-format data can also be visualized with the stand alone (off-line) use of DAWN, giving a g4.prim-format file name from a command line as an argument.

```
% dawn g4.prim-format-filename
```

The g4.prim format is designed with the following philosophy:

1. The format should be suitable to describe 3D data of GEANT4. For this purpose, (i) it should support most of shapes defined in GEANT4 as built-in primitives, and (ii) it should be able to describe both the CSG and B-Rep data.
2. Attributes such as color should be described with the way of state machine in order to avoid repetitive resetting of them.
3. The format should be able to describe rendering commands for DAWN as well as 3D data, but the rendering commands should be clearly distinguishable from 3D data.
4. The format should be suitable for effective data transfer via local or wide area network. For this purpose, it should be compact and be able to be translated into VRML format easily.

2 3D Data and rendering commands

Each 3D data begins with a character '/' (slash). For example, a $1 \times 1 \times 1$ box is described as:

```
/Box  0.5 0.5 0.5
```

where the 0.5's are half lengths of edges. On the other hand, each rendering command for DAWN begins with a character '!' (exclamation mark). For example,

```
!DrawAll
```

is a command to request DAWN to flush drawing. Each 3D data and rendering command need not begin at the first column of the line, i.e., indentation is allowed.

3 Comment lines and blank lines

A Line beginning with a character '#' is regarded as a comment line. The character '#' must be put at the first column. (Indentation of comment lines are not allowed.) The first line of the whole data should be a comment line

as "`##G4.PRIM-FORMAT-2.4`". We call this line the "header-comment line" below.

Blank lines are acceptable except for the very first line of the whole date.

4 Global structure of g4.prim format

The g4.prim format consists of three blocks: (1) preamble (2) modeling block, and (3) terminating block

4.1 Preamble

The preamble is of the following form:

```
##G4.PRIM-FORMAT-2.4
/BoundingBox  xmin ymin zmin xmax ymax zmax
!SetCamera
!OpenDevice
```

The first line of a g4.prim-format data must be the header-comment line as "`##G4.PRIM-FORMAT-2.4`". The line, `/BoundingBox ...` defines a bounding box which determines extension of the described 3D scene. Real numbers `xmin`, `ymin` and `zmin` are minimum x, y, and z world coordinates of the 3D scene, while `xmax`, `ymax` and `zmax` are maximum. DAWN uses this information for automatic camera positioning, automatic drawing of coordinate axes, etc.

Example:

```
##G4.PRIM-FORMAT-2.4
/BoundingBox -1.0 -2.5 -5.0 1.0 2.5 5.0
!SetCamera
!OpenDevice
```

4.2 Modeling block

The modeling block begins with a line `!BeginModeling`, and ends with a line `!EndModeling`. Three kinds of 3D data are described between them.

1. 3D primitives, i.e, shapes and polylines

2. Current body coordinates used to locate 3D primitives
3. Current attributes to be assigned to 3D primitives
4. Markers to be set to arbitrary 3D positions

The available 3D primitives, attributes, and markers are listed later together with a detailed description of their formats.

The current body coordinates are defined by the following two lines:

```
/Origin      Ox  Oy  Oz
/BaseVector  e1.x  e1.y  e1.z  e2.x  e2.y  e2.z
```

where (`Ox`, `Oy`, `Oz`) is the origin of the current body coordinates. Orientation of the current body coordinates is defined with a set of base vectors, i.e., x-directional base vector (`e1.x`, `e1.y`, `e1.z`) and y-directional base vector (`e2.x`, `e2.y`, `e2.z`). Z-directional base vector is defined as (`e1.x`, `e1.y`, `e1.z`) $\times$ (`e2.x`, `e2.y`, `e2.z`). The base vectors need not be normalized. The origin and the base vectors are expressed in terms of the world coordinates. (There is no concept of hierarchical coordinate transformation.) The default current body coordinates, which need not be explicitly described are:

```
/Origin      0.0 0.0 0.0
/BaseVector  1.0 0.0 0.0  0.0 1.0 0.0
```

In other words, the default current body coordinates are identical with the world coordinates.

Attributes are able to be assigned to 3D primitives. The most important one is color: `/ColorRGB R G B`. For example, a red box is described by putting a line `/ColorRGB 1.0 0.0 0.0` before a line `/Box ....` Each of R (red), G (green), and B (blue) components of color should take a value between 0 and 1. The default color which need not be explicitly described is white, which is equivalent to `/ColorRGB 1.0 1.0 1.0`. Once current body coordinates or current attributes are set, they remain effective until they are explicitly described later again with other values.

Example: Red $1 \times 4 \times 9$ box centered at position (1, 2, 3)

```
!BeginModeling
/Origin   1  2  3
/ColorRGB 1  0  0
/Box 0.5   2.0   4.5
!EndModeling
```

4.3 Terminating block

The terminating block describes a few rendering commands as follows.

```
!DrawAll
!CloseDevice
```

These commands request DAWN to flush drawing and close the current visualizing device.

4.4 A Real example of complete visualizable file

The following is a visualizable example of a red $1 \times 4 \times 9$ box centered at position (1,2,3) of the world coordinate.

```
##G4.PRIM-FORMAT-2.4
#####################################
# Red 1x4x9 box centered at (1,2,3) #
#####################################
/BoundingBox  0.5  0.0  -1.5   1.5   4.0   7.5
!SetCamera
!OpenDevice
!BeginModeling
/Origin   1.0  2.0  3.0
/ColorRGB 1.0  0.0  0.0
/Box 0.5   2.0   4.5
!EndModeling
!DrawAll
!CloseDevice
```

5 Formats of 3D primitives

In this section, we explain formats of available 3D primitives, i.e., shapes and polylines, one by one. Note that current body coordinates are used in the description. For example, the x-axis means the one in the current body coordinates. All angles are described in units of radians.

5.1 /Box

`/Box` describes a box with edges parallel to the x, y, and z axes. Its center locates at the origin. `/Box` corresponds to class G4Box of GEANT4.

Format	/Box dx dy dz
dx	half length of x-directional edges
dy	half length of y-directional edges
dz	half length of z-directional edges

5.2 /Column

/Column describes a solid cylinder. Its height extends along the z axis. The top circle is on plane z = +dz, and the bottom circle on plane z = -dz. Centers of the top and bottom circles are (0, 0, +dz) and (0, 0, -dz), respectively. /Column is equivalent to /Tubs with full azimuthal angle (2π) and zero minimum radius.

Format	/Column r dz
r	radius of the top and bottom circles
dz	half height along the z axis

5.3 /Cons

/Cons describes a tube or its segment in azimuthal angle with varying minimum (inside) and maximum (outside) radii. It is a cone (segment) with its upper part cut away. Its height extends along the z axis. The top facet is on plane z = +dz, and the bottom facet on plane z = -dz. Centers of the top and bottom facets are (0, 0, +dz) and (0, 0, -dz), respectively. /Cons corresponds to class G4Cons of GEANT4.

Format	/Cons rmin1 rmax1 rmin2 rmax2 dz sphi dphi
rmin1	minimum (inside) radius at z = -dz
rmax1	maximum (outside) radius at z = -dz
rmin2	minimum (inside) radius at z = +dz
rmax2	maximum (outside) radius at z = +dz
dz	half height along the z axis
sphi	starting azimuthal angle, $[-2\pi, +2\pi]$
dphi	extension of azimuthal angle, dphi = $[0, 2\pi]$, sphi + dphi = $[-2\pi, +2\pi]$

5.4 /Parallelepiped

`/Parallelepiped` describes a parallelepiped, which is the following skewed box.

1. The top and bottom facets are identical parallelograms.
2. The top parallelogram is on plane z = +dz and the bottom parallelogram on plane z = -dz.
3. The center locates at the origin.
4. A line joining the centers of the top and bottom parallelograms is skewed by angles θ (polar angle) and ϕ (azimuthal angle).
5. Two edges of the top (or bottom) parallelogram are parallel to the x axis. Their length is 2× `dx`, and distance between them, i.e., height of the parallelogram, is 2× `dy`.
6. The top (or bottom) parallelogram skews by angle α in the x direction. That is, α is the angle formed the y axis and a line joining middle points of the two x-directional edges mentioned above.

`/Parallelepiped` corresponds to class G4Para of GEANT4.

Format	`/Parallelepiped` `dx dy dz`
	`tanAlpha`
	`tanTheta_cosPhi`
	`tanTheta_sinPhi`

`dx`	half length of edges parallel to the x axis
`dy`	half height of the top and bottom parallelograms along the y axis
`dz`	half height of this parallelepiped along the z axis
`tanAlpha`	$\tan(\alpha)$ α: angle expressing skew of the top and bottom parallelograms in the x direction.
`tanTheta_cosPhi`	$\tan(\theta) \times \cos(\phi)$ Polar angle θ and azimuthal angle ϕ describes skewness of the line joining centers of the top and bottom parallelograms
`tanTheta_sinPhi`	$\tan(\theta) \times \sin(\phi)$

5.5 /PolyCone

`/PolyCone` describes a rotational body around the z axis. The shape of `/PolyCone` is piled-up `/Cons`'s in the z direction. `/PolyCone` corresponds to class G4BREPSolidPCone of GEANT4. But `/PolyCone` is not really used in GEANT4 visualization. `/Polyhedron ... /EndPolyhedron` is used instead.

Format	`/PolyCone sphi dphi nz`
	`z[nz] rmin[nz] rmax[nz]`
	(A[n] abbreviates a list of n real numbers)

`sphi`	starting azimuthal angle, $[-2\pi, +2\pi]$
`dphi`	extension of azimuthal angle, `dphi` = $[0, 2\pi]$ and `sphi` + `dphi` = $[-2\pi, +2\pi]$
`nz`	number of given z coordinates to define piled-up `/Cons`'s: `nz` is equal to the number of the piled-up `/Cons`'s plus one.
`z[i]`	z coordinate of the top plane of the i-th `/Cons`. The `z[0]` defines the bottom plane of the whole shape. i = 0,1,2, ..., `nz` -1.
`rmin[i]`	minimum (inside) radius at z = z[i]
`rmax[i]`	maximum (outside) radius at z = z[i]

5.6 /PolyGon

`/PolyGon` is similar to `/PolyCone`. The difference is that its side is not a curved surface, but its side is really a set of explicitly given numbers of 3D polygons. `/PolyGon` corresponds to class G4BREPSolidPolyhedra of GEANT4. But `/PolyGon` is not really used in GEANT4 visualization. `/Polyhedron ... /EndPolyhedron` is used instead.

Format	`/PolyGon sphi dphi nsides nz`
	`z[nz] rmin[nz] rmax[nz]`
	(A[n] abbreviates a list of n real numbers)

`nsides`	accuracy of expressing curved side surfaces
others	same as `/PolyCone`

5.7 /Polyhedron ... /EndPolyhedron

`/Polyhedron` ... `/EndPolyhedron` describes a general polyhedron composed of a set of 3D polygons. Concave polygons are available but not recommended. Lists of vertices and facets are described between `/Polyhedron` and `/EndPolyhedron`. The following is the format for a polyhedron with N vertices and M facets.

Format	`/Polyhedron`
	`/Vertex x1 y1 z1`
	`/Vertex x2 y2 z2`
	`.....`
	`/Vertex xN yN zN`
	`/Facet f1_v1 f1_v2 f1_v3 ...`
	`/Facet f2_v1 f2_v2 f2_v3 ...`
	`.....`
	`/Facet fM_v1 fM_v2 fM_v3 ...`
	`/EndPolyhedron`

`(xi, yi, zi)`	3D position of the i-th vertex of the polyhedron ($1 \leq$ `i` $\leq N$). The integer label 'i" is assigned to vertices incrementally, beginning from 1.
`fa_vj`	the j-th vertex label of the a-th facet, whose absolute value must coincide with one of the vertex labels $\{1, 2, 3, \ldots, N\}$.
`fa_v1 fa_v2 fa_v3 ...`	the a-th facet expressed by connecting vertex labels in counter-clock-wise order viewing from its front side. Edge `fa_vi`—`fa_vj` is regarded as an invisible edge if `fa_vj` is negative. At least three vertices must be described.

Note that positive-integer labels are incrementally and automatically assinged to vertices given by `/Vertex ...`, beginning from 1, not 0.

An invisible edge is expressed by assigning a negative integer vertex label to its ending point. For example, "/Facet 1 -2 6 5 ;" describes a facet 1→2→6→5→1, in which the edge connecting vertices 1 and 2 is invisible.

Vertex data (`/Vertex ...`) and facet data (`/Facet ...`) can be described at any position with any order between `/Polyhedron` ... `/EndPolyhedron`,

but it is recommended that all vertex data are described beforehand.

The following is a sample description of a red $1 \times 4 \times 9$ box using `/Polyhedron` ... `/EndPolyhedron`.

Example: 1×4 × 9 box

```
##G4.PRIM-FORMAT-2.4
#####################################################
# 1x4x9 box with /polyhedron ... /EndPolyhedron ##
#####################################################

/BoundingBox -0.5 -2.0 -4.5  0.5  2.0  4.5
!SetCamera
!OpenDevice
!BeginModeling

/ColorRGB  1.0 0.0 0.0

/Polyhedron
/Vertex -0.500000000 -2.000000000 -4.500000000
/Vertex  0.500000000 -2.000000000 -4.500000000
/Vertex -0.500000000 2.000000000 -4.500000000
/Vertex  0.500000000 2.000000000 -4.500000000
/Vertex -0.500000000 -2.000000000 4.500000000
/Vertex  0.500000000 -2.000000000 4.500000000
/Vertex -0.500000000 2.000000000 4.500000000
/Vertex  0.500000000 2.000000000 4.500000000
/Facet 3 4 2 1
/Facet 1 2 6 5
/Facet 2 4 8 6
/Facet 4 3 7 8
/Facet 3 1 5 7
/Facet 5 6 8 7
/EndPolyhedron


!EndModeling
!DrawAll
!CloseDevice
```

5.8 /Polyline ... /EndPolyline

`/Polyline ... /EndPolyline` describes a polyline, i.e., a set of successive line segments. A List of vertices is described between `/Polyline` and `/EndPolyline`. The following is the format of describing a polyline with N vertices, i.e., $N-1$ line segments.

Format	`/Polyline`
	`/PLVertex x1 y1 z1`
	`/PLVertex x2 y2 z2`
	`.....`
	`/PLVertex xN yN zN`
	`/EndPolyline`

`(xi, yi, zi)`	the i-th vertex position

5.9 /Sphere

`/Sphere` describes a sphere. Its center locates at the origin.

Format	`/Sphere r`
`r`	radius

5.10 /SphereSeg

`/SphereSeg` describes a sphere segment in polar and/or azimuthal angles. It corresponds to class G4Sphere of GEANT4.

Format	`/SphereSeg rmin rmax stheta dtheta sphi dphi`
`rmin`	minimum (inside) radius
`rmax`	maximum (outside) radius
`stheta`	starting polar angle, $[0, \pi]$
`dtheta`	extension of polar angle, `dtheta` $= [0, \pi]$ and `stheta` + `dtheta` $= [0, \pi]$
`sphi`	starting azimuthal angle, $[-2\pi, +2\pi]$
`dphi`	extension of azimuthal angle, `dphi` $= [0, 2\pi]$ and `sphi` + `dphi` $= [-2\pi, +2\pi]$

5.11 /Torus

`/Torus` describes a torus or its segment in azimuthal angle around the z axis. The curbed tube to form the torus has a constant minimum inside radius (rmin) and a maximum outside radius (rmax). The circle drawn by the central line of the tube has a radius rtor, and center of this circle is the coordinate origin. `/Torus` corresponds to class G4Torus of GEANT4.

Format	`/Torus rmin rmax rtor sphi dphi`
`rmin`	minimum (inside) radius of tube
`rmax`	maximum (outside) radius of tube
`rtor`	radius of circle drawn by the central line of the tube
`sphi`	starting azimuthal angle of the above circle, `sphi` $= [-2\pi, +2\pi]$
`dphi`	extension of azimuthal angle of the above circle, `dphi` $= [0, 2\pi]$, `sphi` + `dphi` $= [-2\pi, +2\pi]$

5.12 /Trap

`/Trap` is a skewed version of `/Trd`, i.e., asymmetric pyramid with its upper part cut away. Its top and bottom facets are asymmetric trapezoids. Its skew is expressed with direction of a line joining the centers of the top and bottom trapezoids. (Here we define a center of a trapezoid by an intersection point of two lines passing through middle points of opponent edges.) This line should pass through the origin. The top trapezoid is on plane z = +dz, and the bottom trapezoid on plane z = -dz. Note that there are 11 parameters, but only 9 are really independent. Meanings of some parameters are similar to those of `/Parallelepiped`. `/Trap` corresponds to class G4Trap of GEANT4.

Format	/Trap dz theta phi h1 bl1 tl1 alpha1 h2 bl2 tl2 alpha2
dz	half height of this shape along the z axis
theta	polar angle of the line expressing skew
phi	azimuthal angle of the line expressing skew
h1	half height of the bottom trapezoid along the y axis
bl1	half length along the x direction of the side at minumum y of the bottom trapezoid
tl1	half length along the x direction of the side at maximum y of the bottom trapezoid
alpha1	angle formed the y axis and a line joining middle points of the two x-directional edges of the bottom trapezoid
h2	half height of the top trapezoid along the y axis
bl2	half length along the x direction of the side at minimum y of the top trapezoid
tl2	half length along the x direction of the side at maximum y of the top trapezoid
alpha2	angle formed the y axis and a line joining middle points of the two x-directional edges of the bottom trapezoid

5.13 /Trd

`/Trd` describes a symmetric pyramid with its upper part cut away. Its properties are:

1. The top and bottom facets are rectangles with, in general, different sizes.
2. Two edges of the top (or bottom) rectangles are parallel to the x axis, and the other two edges are parallel to the y axis.
3. The top rectangle is on plane z = +dz, and the bottom rectangle on plane z = -dz.
4. Centers of the top and bottom rectangles are (0, 0, +dz) and (0, 0, -dz), respectively.

`/Trd` corresponds to class G4Trd of GEANT4.

Format	/Trd dx1 dx2 dy1 dy2 dz
dx1	half length of x-parallel edges at z = -dz
dx2	half length of x-parallel edges at z = +dz
dy1	half length of y-parallel edges at z = -dz
dy2	half length of y-parallel edges at z = +dz
dz	half height along the z axis

5.14 /Tubs

/Tubs describes a tube or its segment in azimuthal angle with constant minimum (inside) and maximum (outside) radii. Its height extends along the z axis. The top facet is on plane z = +dz, and the bottom facet on plane z = -dz. Centers of the top and bottom facets are (0, 0, +dz) and (0, 0, -dz), respectively. /Tubs is equivalent to /Cons with rmin1 = rmin2 and rmax1 = rmax2. /Tubs corresponds to class G4Tubs of GEANT4.

Format	/Tubs rmin rmax dz sphi dphi
rmin	minimum (inside) radius
rmax	maximum (outside) radius
dz	half height along the z axis
sphi	starting azimuthal angle, $[-2\pi, +2\pi]$
dphi	extension of azimuthal angle, dphi = $[0, 2\pi]$, sphi + dphi = $[-2\pi, +2\pi]$

6 Formats of attributes

This section describes the formats of available attributes. Each attribute has an current value, which is assigned to described 3D primitives automatically.

6.1 /ColorRGB

/ColorRGB describes current color. For 3D shapes such as /Box, it is used as a set of diffusion reflection coefficients. For polylines, it is regarded as their color directly. By default, /ColorRGB 1.0 1.0 1.0 is set implicitly.

Format	/ColorRGB R G B
(R, G, B)	red, green, and blue components of color. R, G, B = [0,1]

6.2 /FontName

`/FontName` sets a current font used by `/MarkText2D` and `/MarkText2DS`, and `/Text2DS`. Its argument should coincide with one of a font name supported by a PostScript driver used for printing. Greek letters are also available by setting the argument to "Symbol". By default, `/FontName Times-Roman` is set implicitly.

Format	`/FontName fontname`
fontname	string to specify font name (e.g. Times-Roman, Courier, Symbol)

6.3 /ForceWireframe

`/ForceWireframe` is a boolean flag, which describes current forcible wireframe mode. Its value is either 0 or 1. If it is set to 1, 3D shapes are visualized with wireframe style forcibly. If it is set to 0, 3D shapes are visualized according to a drawing style given on the GUI panel of DAWN. By default, `/ForceWireframe 0` is set implicitly.

Format	`/ForceWireframe flag`
`flag`	1: forcible wireframe mode is made active 0: forcible wireframe mode is made inactive

`/ForceWireframe` is often used to make a shape look transparent. For example, the following lines described in the modeling block displays a green opaque sphere put in a red transparent box:

```
### green opaque sphere (inside)
/ColorRGB 0.0  1.0  0.0
/ForceWireframe 0
/Sphere     1.0
### red transparent box (outside)
/ColorRGB 1.0  0.0  0.0
/ForceWireframe 1
/Box 1.1  1.1 1.1
```

DAWN does not support a way of expressing half-transparent features of shapes.

6.4 /Ndiv

`/Ndiv` describes current accuracy in approximating a curved surface with a set of polygons. For `/Column`, `/Cons`, `/PolyCone`, `/PolyGon`, and `/Tubs`, each of their side surfaces is expressed with `/Ndiv` small rectangles. For `/Sphere`, its surface is expressed with 2× `/Ndiv` × `/Ndiv` small rectangles or triangles. By default, `/Ndiv 24` is set implicitly.

Format	`/Ndiv n`
`n`	accuracy of approximating curved surfaces, n $\geq$ 3

7 Formats of markers

Visible markers can be set to arbitrary 3D positions. They correspond to classes inherited from G4VMarker in GEANT4. The ending string "2D" or "2DS" of each marker name, e.g. `/MarkSquare2D` and `/MarkSquare2DS`, means that every marker always shows its front face towards the view point, and so, for example, a marker `/MarkSquare2D` always looks as right square. The marked 3D positions are expressed with the current body coordinates. Colors are decided by the current color attribute. As for marker size, there are some points to be well understood in use:

1. A unit of marker size is either the real 3D-world unit or the 2D unit on screen:
   (a) Size of a marker whose name ends with character 'S', e.g. `/MarkCircle2DS`, is described with the 2D unit on screen.
   (b) Size of a marker whose name does not end with character 'S', e.g. `/MarkCircle2D` is described with the 3D unit.
2. The 2D unit is recognized as "pixel" on computer displays and "pt" (1[pt]= (25.4/72)mm) on printed out papers.
3. For `/MarkText2D` and `/MarkText2DS`, their marker sizes mean font sizes.

Note that in GEANT4 sizes of markers are given as diameters, while in g4.prim-format data they are given as radii.

An important feature of markers is that they are always drawn at the end of visualization, such that they can always become visible. In other words, hidden surface elimination is not applied to markers.

7.1 /MarkCircle2D

`/MarkCircle2D` describes a marker of a two-dimensional circle put in the 3D world. `/MarkCircle2D` corresponds to class G4Circle of GEANT4 with its size given as world-coordinate size.

Format	`/MarkCircle2D  x y z  r`
(x, y, z)	marked 3D position
r	marker size (3D unit): radius of the circle

7.2 /MarkCircle2DS

`/MarkCircle2DS` describes a marker of a two-dimensional circle put in the 3D world. `/MarkCircle2DS` corresponds to class G4Circle of GEANT4 with its size given as screen size.

Format	`/MarkCircle2DS  x y z  r`
(x, y, z)	marked 3D position
r	marker size (2D unit): radius of the circle

7.3 /MarkSquare2D

`/MarkSquare2D` describes a marker of a two-dimensional right square put in the 3D world. `/MarkSquare2D` corresponds to class G4Square of GEANT4 with its size given as world-coordinate size.

Format	`/MarkSquare2D  x y z  dx`
(x, y, z)	marked 3D position
dx	marker size (3D unit): half edge length of the square

7.4 /MarkSquare2DS

`/MarkSquare2D` describes a marker of a two-dimensional right square put in the 3D world. `/MarkSquare2DS` corresponds to class G4Square of GEANT4 with its size given as screen size.

Format	`/MarkSquare2DS  x y z  dx`
(x, y, z)	marked 3D position
dx	marker size (2D unit): half edge length of the square

7.5 /MarkText2D

`/MarkText2D` displays a text (string) to a marked 3D position. The marked position is the left end of the text. `/MarkText2D` corresponds to class G4Text of GEANT4 with its size given as world-coordinate size.

Format	`/MarkText2D x y z size x_offset y_offset string`
(x, y, z)	marked 3D position (left end of string)
size	font size (3D unit)
x_offset	horizontal offset (3D unit)
y_offset	vertical offset (3D unit)
string	string to be displayed (Do not quote with “”.)

7.6 /MarkText2DS

`/MarkText2DS` displays a text (string) to a marked 3D position. The marked position is the left end of the text. `/MarkText2DS` corresponds to class G4Text of GEANT4 with its size given as screen size.

Format	`/MarkText2DS x y z size x_offset y_offset string`
(x, y, z)	marked 3D position
size	font size (2D unit, i.e. pt)
x_offset	horizontal offset (2D units, i.e. pt)
y_offset	vertical offset (2D units, i.e. pt)
string	string to be displayed (Do not quote with “”.)

For example, suppose that we want to display a string "Fukui Renderer DAWN" with the following attributes:

- position: (1,2,3) in the current body coordinate
- color : red
- font name : Courier
- font size : 12 (2D unit)
- horizontal offset: 6 (2D unit)
- vertical offset: 3 (2D unit)

The following lines described in the modeling block displays this string on the screen.

```
/ColorRGB  1 0 0
/FontName Courier
/MarkText2DS 1 2 3 12 6 3 Fukui Renderer DAWN
```

Note that all arguments beginning with the 7-th argument, i.e., “Fukui” in the above example, are grouped and treated as one string.

8 Drawing Texts directly on 2D screen

8.1 /Text2DS

`/Text2DS` displays a text (string) directly on the two-dimensional space on screen. No corresponding classes exist in GEANT4. Note that its position, i.e., left end of the string, is described with 2D position vector described in units of “mm”, and its origin is the left bottom of the screen. Font size should be given in units of “pt”. `/Text2DS` is convenient to add title strings etc to drawn figures afterwards.

Format	`/Text2DS  x_mm y_mm  size_pt string`
(x_mm, y_mm)	Left end of string on screen. Unit is mm. Origin (0,0) is the left-bottom.
size	font size (in units of pt)
string	string to be displayed

For example, the following lines described in the modeling block displays a large title ”Supported Primitives 1” at the bottom center of the screen:

```
### black color
/ColorRGB 0 0 0
### font name
/FontName Times-Roman
### 40-pixel string at (30, 40) on screen
/Text2DS 30 40 40 Supported Primitives 1
```

9 Sample g4.prim-format data files

Many viewable sample g4.prim-format data files are included in the DAWN package. For example, see files `primitives.prim` and `primitives2.prim`.