Author: Linda Jun

Created: September 30, 2013

Proposed Color Implementation in GMAT

1. Purpose

The purpose of this document is to design color so that it will be easier to represent color in GMAT. Currently color is represented using unsigned integer numbers which is hard to use in a script. There are two issues related to color need to resolved.

1. Issues related to color:

**GMT-2447 More user-friendly color input** - In the script, there should be a better way of inputting colors. The current way (BGR, in decimal) is non-standard. A better choice would be to allow color names or an RGB hex value.

**GMT-3825 Per-segment orbit colors** - Add the ability to assign different orbit colors in OrbitView for different Propagate segments in the same trajectory.

1. Current Color Implementation In GMAT

In GMAT, currently color is implemented in base/util/RgbColor class.

class GMAT\_API RgbColor

{

public:

RgbColor();

RgbColor(const Byte red, const Byte green, const Byte blue, const Byte alpha = 0);

RgbColor(const UnsignedInt rgb);

RgbColor(const RgbColor &copy);

RgbColor& operator=(const RgbColor &right);

virtual ~RgbColor();

UnsignedInt GetIntColor();

Byte Red();

Byte Green();

Byte Blue();

Byte Alpha();

void Set(const Byte red, const Byte green, const Byte blue, const Byte alpha = 0);

void Set(const UnsignedInt rgb);

// New static methods needed to be added

static UnsignedInt GetPredefinedColor(const std::string &colorName);

static std::string GetRgbValueString(const UnsignedInt rgb); // return 3 element vector string

static UnsignedInt GetIntColor(const std::string &colorName);

static std::string GetColorName(const UnsignedInt rgb);

static std::string GetColorName(const std::string &rgbString);

private:

struct RgbType

{

unsigned red : 8;

unsigned green : 8;

unsigned blue : 8;

unsigned alpha : 8;

} rgbType;

union ColorType

{

RgbType rgbColor;

UnsignedInt intColor;

} colorType;

};

There are predefined colors are defined in base/util/ColorTypes.hpp in namespace GmatColor. More predefined colors may be added to GmatColor as needed.

namespace GmatColor

{

// Value should be in 0x00BBGGRR

const UnsignedInt AQUA = 0x00ffff00;

const UnsignedInt BEIGE = 0x00b5f2fd;

const UnsignedInt BLACK = 0x00000000;

const UnsignedInt BLUE = 0x00ff0000;

const UnsignedInt CHESTNUT = 0x00404080;

const UnsignedInt FUCHSIA = 0x00ff00ff;

const UnsignedInt GOLDTAN = 0x00005E5E;

const UnsignedInt GRAY = 0x00808080;

const UnsignedInt GREEN = 0x00008000;

const UnsignedInt LIME = 0x0000ff00;

const UnsignedInt NAVY = 0x00800000;

const UnsignedInt ORANGE = 0x004080ff;

const UnsignedInt PINK = 0x00d9b6aa;

const UnsignedInt PURPLE = 0x00800080;

const UnsignedInt RED = 0x000000ff;

const UnsignedInt SILVER = 0x00c0c0c0;

const UnsignedInt SKYBLUE = 0x00930000;

const UnsignedInt TEAL = 0x00808040;

const UnsignedInt YELLOW = 0x0000ffff;

const UnsignedInt WHITE = 0x00ffffff;

const UnsignedInt D\_BLUE = 0x00780000;

const UnsignedInt D\_BROWN = 0x001b588b;

const UnsignedInt D\_GRAY = 0x00404040;

const UnsignedInt D\_TEAL = 0x00414121;

const UnsignedInt L\_BLUE = 0x00dbb767;

const UnsignedInt L\_BROWN = 0x001a98ce;

const UnsignedInt L\_CHESTNUT = 0x00303060;

const UnsignedInt L\_GRAY = 0x00e0e0e0;

};

1. Proposed design for loading and saving color

Currently in GMAT, colors are read in as Unsigned Integer value. I propose to use predefined color names and/or RGB value in Unsigned Byte Vector form in scripting. The Unsigned Byte represents value between 0 and 255. For example a vector [255, 0, 0] represents Red. A vector [0, 0, 0] represents Black and [255, 255, 255] represents White. As using predefined color name is limited, using RGB value gives more flexible in representing colors. A User can easily read RGB values from the wxColourDialog which GMAT currently using for choosing colors from the OrbitView and GroundTrackPlot panels. The downside of using RGB value is that it is hard to figure out the color when it is used in a script. The design assumes that we want to use both Color Names and RGB values in scripting. Internally all colors are stored as UnsignedInt. All range checking should be done for both color names and RGB value when loading a script.

Question: Do we want to use only RGB value for easier implementation? Or do we want to implement RGB color first and then do the Color Name later?

The OrbitView and GroundTrackPlot draws trajectory of SpacePoints during the run. To implement the color for SpacePoint objects such as CelestialBody, Spacecraft, GroundStation, Barycenter, and LibrationPoint, the color information will be stored in the SpacePoint class. Each SpacePoint subclass assigns the default color in the constructor. The SpacePoint class will be modified as follows:

class GMAT\_API SpacePoint : public GmatBase

{

public:

// New access methods for color

// Parses color name or RGB value and store as UnsignedInt color

virtual std::string GetStringParameter(const Integer id) const;

virtual bool SetStringParameter(const Integer id,

const std::string &value);

virtual std::string GetStringParameter(const std::string &label) const;

virtual bool SetStringParameter(const std::string &label,

const std::string &value);

…

protected:

…

enum

{ …

ORBIT\_COLOR // new STRING\_TYPE field

TARGET\_COLOR // new STRING\_TYPE field

SpacePointParamCount

};

…

UnsignedInt orbitColor; // new member

UnsignedInt targetColor; // new member

}

The XYPlot plots Parameters during the run. The color information for Parameters such as Calculated Parameters, Variables, and ArrayElements is currently stored in the base Parameter class. Each Parameter subclass currently assigns the default color in the constructor. The Parameter class will be modified as follows to add a couple of new fields.

class GMAT\_API Parameter : public GmatBase

{

// New access methods for color

// Parses color name or RGB value and store as UnsignedInt color

virtual std::string GetStringParameter(const Integer id) const;

virtual bool SetStringParameter(const Integer id,

const std::string &value);

virtual std::string GetStringParameter(const std::string &label) const;

virtual bool SetStringParameter(const std::string &label,

const std::string &value);

…

protected:

bool mUseDefaultColor; // new flag indicates to use Parameter default color

UnsignedInt mColor; // existing member

UnsignedInt mTargetColor; // new field to plot in different color during solver iteration

}

Currently orbit and target colors are configurable in OrbitView and GroundTrackPlot. Those colors are stored in the parent OrbitPlot class. The Subscriber class will be changed to store the color information such as use default or new color.

Question: Do we want to configure colors for XYplot?

class GMAT\_API Subscriber : public GmatBase

{

public:

…

Void SetOrbitColor(bool useDefault, UnsignedInt newColor);

…

protected:

…

bool mUseDefaultColor; // new field to set back to Parameter or SpacePoint default color

UnsignedInt mOrbitColor; // new orbit color set from Propagate command

…

}

1. Loading colors from a script

The following shows proposed scripting for setting colors for SpacePoint objects. If color field is not present, it will use the default color assigned in each subclass such as Spacecraft, Earth, and LibrationPoint, etc. If color fields are not present, it will use the predefined color from the object.

Using color name:

Sat.OrbitColor = Red; % All color names are case insensitive

Sat.TargetColor = Teal;

Earth.OrbitColor = Green;

Earth.TargetColor = White;

Using color value:

Sat.OrbitColor = [255 0 0]; % Color value of RGB is in unsigned byte value (0 – 255)

Sat.TargetColor = [64 128 128];

Using mixed color type:

Sat.OrbitColor = Red;

Sat.TargetColor = [64 128 128];

The following shows proposed scripting for setting color for OrbitView and GroundTrack plots. A use can set new colors or use default colors from the object. If color fields are not present, it will use the predefined color from the object.

Using default object color:

OrbitView.Add = {Sat, Earth};

OrbitView.OrbitColor = [Default Default]; % Sat color comes from Sat, Earth color comes form Earth

OrbitView.TargetColor = [Default Default];

Overriding colors by color name:

OrbitView.Add = {Sat, Earth};

OrbitView.OrbitColor = {Red Green};

OrbitView.TargetColor = {Teal Gray};

Overriding colors by color value:

OrbitView.Add = {Sat, Earth};

OrbitView.OrbitColor = {[255 0 0] [0 128 0]};

OrbitView.TargetColor = {[64 128 128] [0 128 0]};

Overriding colors by mixed color type:

OrbitView.Add = {Sat, Earth};

OrbitView.OrbitColor = {Red [0 128 0]};

OrbitView.TargetColor = {Teal [0 128 0]};

Using mixed default and overriding colors:

OrbitView.Add = {Sat, Earth};

OrbitView.OrbitColor = {Default [0 128 0]};

OrbitView.TargetColor = {Teal Default};

The following shows proposed scripting for setting colors for XY plot. A use can set new colors or use defaults from Parameter object. Currently in GMAT, all Parameter types such as X, Y, SMA, have hardcoded default colors and the color information is stored in the Parameter class.

XYPlot1.XVariable = Sat.A1ModJulian;

XYPlot1.YVariables = {Sat.EarthMJ2000Eq.X Var1};

XYPlot1.Colors = {[255 0 0] [0 255 0]};

XYPlot1.Colors = {Red Green};

XYPlot1.Colors = {Default Green}; % Use color from Parameter X, Do we need this?

1. Saving color to a script

Saving color in RGB format will be done by reformatting internally stored color in Unsigned Integer value to Unsigned Byte vector form. If a user enters color in Color Name, it will be stored as string value and converted to internal Unsigned Integer. A bit of coding will be involved to preserve the color format (color name or RGB) when saving it to a script, since we want to write it out as originally scripted. For example, the following script uses Default, RGB value, and Color Name, so we want to preserve the color format setting when OrbitView object is saved.

OrbitView.OrbitColor = {Default [0 128 0]};

OrbitView.TargetColor = {Teal Default};

1. Changing colors form GUI

All SpacePoint derived type panels need to show color pickers and a check box for setting it back to default color. The color can be changed only when “Use Default” check box is unchecked. If “Use Default” check box is checked, the color picker will be disabled.

* 1. The following panels are identified for change:
     1. FormationPanel, GroundStationPanel, BarycenerPanel, LibrationPointPanel : Show at the bottom
     2. SpacecraftPanel: Show at the bottom in the Visualization tab
     3. CelestialBodyPanel: Show at the bottom in the Properties tab
  2. OrbitView and GroundTrack plot panels already show color Pickers, but setting back to default object color needs to be added in case users want to change back to the default.
  3. XYPlot panel needs to add color pickers and a check box for setting it back to default color for each selected Parameters. Question: Do we want this capability for XY plot?

1. Proposed design for showing different colors for different propagate segments

In order to show different colors for different propagate segments, the Propagate command must have the color information. There will be only one color assigned for all propagating spacecrafts in a propagate segment. For example, if propagating multiple spacecrafts, they will all have the same orbit color if color is specified. The Propagate command will internally store color in UnsignedInt and will have a color format indicator to be used for saving to script.

Question: Do we want to show XYPlot in different colors for different propagate segments?

1. Loading orbit color from a script

The following shows proposed scripting for Propagate command.

Propagate Prop(Sat) {Sat.ElapsedSecs = 12000.0, Color = Green}; % Use Green

Propagate Prop(Sat1, Sat2) {Sat1.ElapsedSecs = 12000.0, Color = [255 0 0]}; % Use this color for Sat1, Sat2

Propagate Prop(Sat) {Sat.ElapsedSecs = 12000.0, Color = Default}; % Use color from Sat

Propagate Prop(Sat) {Sat.ElapsedSecs = 12000.0}; % Use color from Sat

Propagate Prop(Sat1, Sat2) {Sat.ElapsedSecs = 12000.0}; % Use color from Sat1 and Sat2

1. Changing orbit color from the Propagate Panel

The Propagate panel needs to be changed to have a color picker and a check box for changing back to default color. The color can be changed only if “Use Default” check box is unchecked. The orbit color picker can be added at the top of the panel and set to the color of the first propagating spacecraft since it shows only one color. If default color is used, it will use the colors from the propagating space objects even though only one color is shown. If “Use Default” check box is checked, the color picker will be disabled.

1. How orbit color changes through the GUI
   1. In Spacecraft panel, it will show spacecraft color (default or set by script)
      * 1. If a user changes the color, the new color is saved to the configured Spacecraft object
   2. In OrbitView or GroundTrackPlot panel, it will show the color for each object
      * 1. If a user changes the color, the new color is saved to the configured object
   3. In Propagate panel, it will show the color from the first propagating spacecraft it will use only one color for the propagate segment.
      * 1. If a user changes the color, the color information such as using default or new orbit color is saved to Propagate command.
2. How color works during runtime

When a script is read, all reference objects are set in the final pass of parsing. If a user changes any color from the GUI, the color information is saved to configured objects.

**Initialization:**

Cloned objects and un-cloned command sequence are placed in the Sandbox during the initialization. The Sandbox sets reference objects to each object and initializes it by calling Initialize() method. Next, the Sandbox initializes commands. The following steps are taken during the initialization:

1. OrbitView::Initialize() – Viewing canvas is created at this time
   1. Passes drawing SpacePoint object pointers to Canvas through PlotInterface.
2. Propagate::Initialize()
   1. Adds propagating Spacecraft object pointers found from the object map.
   2. Registers to Publisher

**Execution:**

1. Propagate::Execute()
   1. Sets orbit color through the Publisher
      1. Publisher::SetOrbitColor(GmatBase \*provider, bool useDefault, UnsignedInt newColor)
         1. Sets color for all registered subscribers
      2. OrbitView::SetOrbitColor(bool useDefault, UnsignedInt newColor)
         1. Saves color information
   2. Publish data through the Publisher
      1. Publisher::Publish(data)
         1. Distributes data to all subscribers
      2. OrbitView::Distribute()
         1. Updates data buffer
         2. Builds color array. If using default color, use color from the plotting objects. If using a new color, use saved new color.
         3. Updates GL plot through PlotInterface