RasterGraphic in C++ using Polymorphic Inheritance and RTTI

Purpose: This is a development of 'Raster Graphic with Overloaded Operators'. It works in a very similar way but with the addition of two new classes `SystemMemoryImage` and `GPUMemoryImage`, that are derived from the abstract base class `Image`. It is a console application that holds the `GraphicElement`s of a `RasterGraphic` application (there is no actual graphics in the project) in a `forward_list` class template of unspecified length in dynamic memory. Each `GraphicElement` now holds a `vector` of `Image*` pointers that are the addresses of the `Image` objects that are actually `SystemMemoryImage` or `GPUMemoryImage` objects. Polymorphic inheritance ensures that any `Image*` in the `vector` will call the correct overridden polymorphic function (`Buffersize()` in this case) for the actual object it points to. Also a `SystemMemoryImage` object uses more memory than a `GPUMemoryImage` object because they reside in different memory locations in the computer: in system memory or GPU local memory respectively. A `GPUMemoryImage` object also has a shader file that is a small fragment of code that executes for each pixel in the Image buffer.
Therefore you will use an expression like

```
Images[i]->BufferSize()
```

to cause one of the `Image*` in the `GraphicElement` vector to calculate the buffer size of the `SystemMemoryImage` or `GPUMemoryImage` it points to without needing to identify what type of Image it really is. The polymorphic function that actually executes is the correct one for the object type, selected through the virtual function table of the object.
In places where you need to identify the type of Image (`SystemMemoryImage` or `GPUMemoryImage`), ***but no***

The `RasterGraphic` is a series of `GraphicElement`s held in a `forward_list`. Each GraphicElement holds its list of Image* in a `vector`. There are now two types of `Image`, as detailed above, both subclasses of the abstract base class `Image`. Each Image object contains its Image time which is set by the user. You can:

- Add a new `GraphicElement` to the `RasterGraphic` at a position in the forward list selected by the user
- Delete the first `GraphicElement` in the `RasterGraphic`
- Run the `RasterGraphic` to show the list of `Image` details of each `GraphicElement` one after another at the Image intervals specified by the user when the Image was entered – note that the output counts up the seconds using a timer as was done in previous Raster Graphic programs.
- Quit

Note the following:

- dynamic memory management is done with `new` and `delete`
- input and output is done with `cin` and `cout`
- there is no unused dynamic memory at any time
- `string` objects are used in the `RasterGraphic` and `GraphicElement` classes to hold strings (a `char` array is still used in the `Image` class)
- Release of dynamically allocated memory is done in destructors so there is no resource leak

Example Output

```
MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElement
 3. Run the RasterGraphic
 4. Quit

1
Insert a GraphicElement in the RasterGraphic
Please enter the GraphicElement filename: Graphic_Element_1
Entering the GraphicElement Images (the sets of dimensions and durations)
Please enter the number of Images: 2
Please enter pixel x-width for Image #0 pixel_x:16
Please enter pixel y-width for Image #0 pixel_y:32
Please enter the duration for this Image: 2
Please enter the name for this Image: Image_1
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 1

Please enter pixel x-width for Image #1 pixel_x:64
Please enter pixel y-width for Image #1 pixel_y:32
Please enter the duration for this Image: 3
Please enter the name for this Image: Image_2
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 2
Please enter the file name of the associated GPU Shader: PS_1

This is the first GraphicElement in the list

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElement
 3. Run the RasterGraphic
 4. Quit

1
Insert a GraphicElement in the RasterGraphic
Please enter the GraphicElement filename: Graphic_Element_2
Entering the GraphicElement Images (the sets of dimensions and durations)
Please enter the number of Images: 1
Please enter pixel x-width for Image #0 pixel_x:1024
Please enter pixel y-width for Image #0 pixel_y:768
Please enter the duration for this Image: 1
Please enter the name for this Image: Image_3
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 2
Please enter the file name of the associated GPU Shader: PS_2


MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElement
 3. Run the RasterGraphic
 4. Quit

1
Insert a GraphicElement in the RasterGraphic
Please enter the GraphicElement filename: Graphic_Element_3
Entering the GraphicElement Images (the sets of dimensions and durations)
Please enter the number of Images: 3
Please enter pixel x-width for Image #0 pixel_x:8
Please enter pixel y-width for Image #0 pixel_y:16
Please enter the duration for this Image: 3
Please enter the name for this Image: Image_4
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 1

Please enter pixel x-width for Image #1 pixel_x:256
Please enter pixel y-width for Image #1 pixel_y:128
Please enter the duration for this Image: 2
Please enter the name for this Image: Image_5
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 2
Please enter the file name of the associated GPU Shader: PS_3

Please enter pixel x-width for Image #2 pixel_x:64
Please enter pixel y-width for Image #2 pixel_y:64
Please enter the duration for this Image: 5
Please enter the name for this Image: Image_6
Please enter the type for this Image (1 = SystemMemoryImage, 2 = GPUMemoryImage): 1

There are 2 GraphicElement(s) in the list
Please specify the position, between 0 and 1 to insert after : 0

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicEleme
 3. Run the RasterGraph
 4. Qu

3
RasterGraphic A
Run the RasterGraphic
GraphicElement #0:      fileName = Graphic_Element_1
        Image #0: System Memory Image
        Image name = Image_1; pixel_x = 16, pixel_y = 32, duration =  2
        Counting the seconds for this Image: 1, 2,
        Memory requirements = 4096 bytes

        Image #1: GPU Memory Image. Shader = PS_1
        Image name = Image_2; pixel_x = 64, pixel_y = 32, duration =  3
        Counting the seconds for this Image: 1, 2, 3,
        Memory requirements = 8192 bytes

GraphicElement #1:      fileName = Graphic_Element_3
        Image #0: System Memory Image
        Image name = Image_4; pixel_x = 8, pixel_y = 16, duration =  3
        Counting the seconds for this Image: 1, 2, 3,
        Memory requirements = 1024 bytes

        Image #1: GPU Memory Image. Shader = PS_3
        Image name = Image_5; pixel_x = 256, pixel_y = 128, duration =  2
        Counting the seconds for this Image: 1, 2,
        Memory requirements = 131072 bytes

        Image #2: System Memory Image
        Image name = Image_6; pixel_x = 64, pixel_y = 64, duration =  5
        Counting the
        Memory requirements = 32768 bytes

GraphicElement #2:      fileName = Graphic_Element_2
        Image #0: GPU Memory Image. Shader = PS_2
        Image name = Image_3; pixel_x = 1024, pixel_y = 768, duration =  1
        Counting the seconds for this Image: 1,
        Memory requirements = 3145728 bytes


Output finished

MENU
 1. Insert a GraphicEleme
 2. Delete the first GraphicEleme
 3. Run the RasterGraph
 4. Qu

2
Delete the first GraphicElement from the RasterGraphic
GraphicElement deleted

MENU
 1. Insert a GraphicEleme
 2. Delete the first GraphicEleme
 3. Run the RasterGraph
 4. Qu

3
RasterGraphic A
Run the RasterGraphic
GraphicElement #0:      fileName = Graphic_Element_3
        Image #0: System Memory Image
        Image name = Image_4; pixel_x = 8, pixel_y = 16, duration =  3
        Counting the seconds for this Image: 1, 2, 3,
        Memory requirements = 1024 bytes

        Image #1: GPU Memory Image. Shader = PS_3
        Image name = Image_5; pixel_x = 256, pixel_y = 128, duration =  2
        Counting the seconds for this Image: 1, 2,
        Memory requirements = 131072 bytes

        Image #2: System Memory Image
        Image name = Image_6; pixel_x = 64, pixel_y = 64, duration =  5
        Counting
        Memory requirements = 32768 bytes

GraphicElement #1:      fileName = Graphic_Element_2
        Image #0: GPU Memory Image. Shader = PS_2
        Image name = Image_3; pixel_x = 1024, pixel_y = 768, duration =  1
        Counting the seconds for this Image: 1,
        Memory requirements = 3145728 bytes


Output finished

MENU
 1. Insert a GraphicEleme
 2. Delete t   first GraphicElement
 3. Run the RasterGraph
 4. Qu
```