RasterGraphic Project in C++ using Polymorphic Inheritance and RTTI

Due Time: 23.59, Sat 01 December 2018 **Earnings:** 9% of your final grade

NOTE: Plan to finish a few days early to avoid last minute hardware/software holdups for which no allowance is given.
NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.

Purpose: This is a development of assignment 2. 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`. Like assignment 2, it is a console application that holds the `GraphicElement`s of a `RasterGraphic` application (there is no actual graphics in the assignment) 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 not where polymorphism is appropriate***, use `dynamic_cast<>`.
To focus on the polymorphic aspects of this final assignment, some of the overloaded operators used in Assignment 2 have been removed.

Part of the code is shown on the next page; it is also on Brightspace in a text file that you can copy and paste. Do not change this code. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no new global variables or functions, no new classes, no macros, no defines and no statics**. Your task is to implement, using C++, only the `RasterGraphic`, `GraphicElement` and `Image` class member functions and the global insertion operators and not add any new ones. Everything you write and submit is in the files: RasterGraphic.cpp, GraphicElement.cpp and Image.cpp.

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 assignment 1 onwards.
- Quit

An example of the output of the running application is given at the end. Yours must work identically and produce identical output.

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 (or you lose 30%).

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:
1. you change the supplied code in any way at all (not a single character) - no code added or removed, no macros, no defines, no statics and no additional classes, global functions or variables,
2. it fails to build in Visual Studio 2015,
3. It crashes in normal operation,
4. it doesn't produce the example output.

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to implement the `RasterGraphic, GraphicElement` and `Image` class member functions in their .cpp files.

What to Submit : Use Blackboard to submit this assignment as a plain zip file (**not** RAR or 7-Zip or 9 Zip) containing only RasterGraphic.cpp, GraphicElement.cpp and Image.cpp. The name of the zipped folder **must** contain your name as a prefix so that I can identify it, for example using my name the file would be tyleraAss3CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header.

Before you submit the code,

- check that it builds and executes in Visual Studio 2015 as you expect - if it doesn't build for me, for whatever reason, you get a deduction of at least 60%.
- make sure you have submitted the correct file – if I cannot build it because the file is wrong or missing from the zip, even if it's an honest mistake, you get 0.

It cannot be late – no time at the semester end. Don't send me files as an email attachment – they will get 0.

Supplied code (also in a text file you can copy and paste on BlackBoard). Don't change it.

```
// Image.h
#pragma once

class Image
{
protected:
          int pixel_x;
          int pixel_y;
          int duration;
          char* name;
public:
          Image(int x, int y, int duration, char* name);
          Image(const Image&);
          virtual ~Image()
          {
                    if(name)
                              delete[]name;
          }
          virtual int BufferSize() = 0;
          friend ostream& operator<<(ostream&, Image&);
};
```

```
// SystemMemoryImage.h
#pragma once
```

```
class  SystemMemoryImage : public Image
{
public:
          SystemMemoryImage(int x, int y, int duration, char* name) :Image(x, y, duration, name) {};
          SystemMemoryImage(const SystemMemoryImage& RGMD) :Image(RGMD) {}
          int BufferSize(){return pixel_x*pixel_y * sizeof(double);}
};
```

```
class GPUMemoryImage : public Image
{
          string shader;
public:
          GPUMemoryImage(int x, int y, int duration, char* name, string shader) :Image(x, y, duration, name), shader(shader)
{};
          GPUMemoryImage(const GPUMemoryImage& RGPUMD) :shader(RGPUMD.shader), Image(RGPUMD) {}
          string GetShader() { return shader; }
          int BufferSize() { return pixel_x*pixel_y * sizeof(float); }
};
```

```
// GraphicElement.h
#pragma once

class GraphicElement
{
          string fileName;
          vector<Image*> Images;
public:
          GraphicElement(string s, vector<Image*> d) :fileName(s), Images(d){}
          GraphicElement(const GraphicElement&);
          ~GraphicElement()
          {
                    vector<Image*>::iterator it;
                    for (it = Images.begin(); it != Images.end(); it++)
                              delete *it;
          }
          friend ostream& operator<<(ostream&, GraphicElement&);
};
```

```
//RasterGraphic.h
#pragma once

class RasterGraphic
{
          string name;
          forward_list<GraphicElement> GraphicElements;
public:
          RasterGraphic(string s): name(s){}
          void InsertGraphicElement();
          void DeleteGraphicElement();
          friend ostream& operator<<(ostream& , RasterGraphic&);
};
```

```
// ass3.cpp
#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC     // need this to get the line identification
//_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF); // in main, after local declarations
//NB must be in debug build
#include <crtdbg.h>
#include <iostream>
#include <string>
#include <vector>
#include <forward_list>
using namespace std;

#include "Image.h"
#include "SystemMemoryImage.h"
#include "GPUMemoryImage.h"
#include "GraphicElement.h"
#include "RasterGraphic.h"

bool running = true;

int main(void)
{
          char selection;
          bool running = true;
          RasterGraphic A("A");
          _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF);

          while (running)
          {
                    cout<<    "MENU\n 1. Insert a GraphicElement\n 2. Delete the first GraphicElement\n 3. Run the
RasterGraphic\n 4. Quit\n"<<endl;
                    cin>>selection;

                    switch (selection)
                    {
                    case '1':
                              A.InsertGraphicElement();
                              break;
                    case '2':
                              A.DeleteGraphicElement();
                              break;
                    case '3':
                              cout << A << endl;
```

```
                                break;
                        case '4':
                                running = false;
                                break;
                        default:
                                break;
                        }
                }
                return 0;
}
```

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 GraphicElement
 3. Run the RasterGraphic
 4. Quit

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 seconds for this Image: 1, 2, 3, 4, 5,
        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 GraphicElement
 2. Delete the first GraphicElement
 3. Run the RasterGraphic
 4. Quit

2
Delete the first GraphicElement from the RasterGraphic
GraphicElement deleted

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

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 the seconds for this Image: 1, 2, 3, 4, 5,
        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 GraphicElement
 2. Delete the first GraphicElement
 3. Run the RasterGraphic
 4. Quit
```