PaintShop Instructions for the Group Project

1. Technical Programing Requirements

For the technical requirements for the PaintShop program, please refer to the PaintShop Requirements Document and Weekly Writing PDF that was posted to Moodle on Monday 2/9 in time for you to get started right away on your Weekly Writing assignment.

2. Group Handin Requirements

As you plan for working on PaintShop as a group. Here are some additional details on what your group will handin to be graded as part of the PaintShop project. There are 2 handin submissions in total:

Sun 2/15, 11:55pm	Team Policies and Expectations, handed in on Moodle
Sun 3/1, 11:55pm	Your PaintShop Program, which includes: 1. The source code for your program, 2. Your Group Design Document (3-page PDF).

(Reminder: No late work will be accepted. We will post our solution to the assignment online the day after it is due.)

Read the sections below carefully for instructions on each of the handins.

2.1 Team Policies and Expectations

This is the document that you started with your team in lab on Fri 2/13. Please finish this with your team and turn it in via the link on Moodle at the normal time for lab work – by Sunday 2/15 at 11:55pm.

2.2 Your PaintShop Program

Your group's PaintShop program is due on Sunday 3/1. There are two handins for the PaintShop program: 1. The actual source code, 2. A 3-page document that describes key design decisions that you made in your code. Expectations for each of these are described in detail here.

2.2.1 Source Code for Your PaintShop Program

Again, please refer back to the PaintShop Requirements Document for specific programming requirements. We will refer specifically to these requirements in our grading.

To turn in your source code, upload a compressed zip file of your PaintShop source code to Moodle, containing a makefile which builds your project on the CSE Labs UNIX computers. This compressed zip file should include all the files in your PaintShop project that are necessary to build your code, including the glui source code. It will also contain your Group Design Document (see section 2.2.2). Upload the single .zip file to the link provided on Moodle by Sunday 3/1 at 11:55pm.

Note: You must ensure that your code will successfully build and run on the CSE Labs UNIX computers when you submit your source code. This will be vital for grading. Also, you may use GitHub's "Download ZIP" button to produce your submission .zip file.

2.2.2 Group Design Document (3-page PDF)

Since this course is as much about good program design as it is about implementing a working program, we want you to tell us about your design. What aspects of the design are you most proud of? What is the most interesting aspect of your design? What is the most controversial, i.e. what decisions did your group debate the most before settling on a final design? We also want you to report a quick indication of how well your group functioned as a team during this iteration of the project.

To convey this to us, you should handin a Group Design Document that is exactly 3 pages long and adheres to the specific formatting requirements described below.

To submit your Group Design Document, place a file called iteration1-designdoc.pdf in the PaintShop/documentation directory before you compress the PaintShop directory.

Page 1 of the Group Design Document:

This is a cover page that should have the following 4 bits of information:

1. Names of group members.

2. Name of github group repository.

3. Statement of completeness of the solution. Ideally, this will just be a single sentence saying that you believe that you have completely and correctly implemented PaintShop according to the specifications provided. If your implementation does not work entirely as intended, then please list what works and what does not work to help us understand how far you got in the project and give you as much credit as possible for the work that you completed.

4. A self-assessment of contributions to the group by each member. Here's how this works: Your group gets 3 stars (*) for each member of your team. If you have a 3-person team, this means you have 9 stars in total. If you have a 4-person team,

you have 12 stars in total. The stars belong collectively to the whole team. At the end of the iteration, you get to give these stars to individual team members as a reward. You have to give out all the stars, but you don't have to give an equal number to each group member. Give as many as you think that group member earned given his/her effort on this iteration of the project.

Here's an example. Peter, Paul, and Mary are a 3-member team. In general, they worked well together and followed the expectations and policies that they agreed upon at the beginning of the project. They each took on slightly different roles, but they each contributed roughly equally to the success of the team. In their report, they can simply write their self-assessment as:

Peter	***
Paul	***
Mary	***

Another example: John, Paul, George, and Ringo are a 4-member team. They started off strong and had a great first meeting during Friday's lab. Ringo offered to take the lead on preparing the PDF handin; John, Paul, and George thought this was a great idea because they were eager to spend most of their time learning hands-on C++ programming. They all agreed to meet as a team in person on Tuesday and Thursday nights. The team made some progress in week 1. In week 2, Ringo got busy with other classes and did not show up for the scheduled meetings. He also responded so slowly to emails that he was basically out of the loop for any important programming decisions. At the end, he showed up to try to do his part by creating the PDF handin, but by that point he was so behind that he didn't really have the knowledge to create this document on his own, so John and Paul did it for him. The team discusses the situation, and they divide the stars up as shown below. Ringo is not too happy about this. He was honestly overloaded with responsibilities for other classes, but the other team members point out that he agreed to a plan at the beginning and he let them down. They don't want this to happen again! And, they are obligated to report this via the star system. Ultimately, Ringo can't be too upset because this is essentially a just a warning; however, he knows that he needs to do a better job contributing to the team in the next iteration of the project.

John	****
Paul	****
George	***
Ringo	*

> The star system (and the discussion that it forces you to have about how well your team has functioned) is intended to be most useful to you (your group) not us (the instructors). Following the policy in our syllabus, the instructors will use this information simply to get a broad sense of how well each group is functioning as a team. Be honest, this self-assessment is a concrete way for you to critique your group. It will not affect your grade, but if someone in your group is not contributing equally, then this is a very clear way to tell him/her and us.

Page 2 of the Group Design Document:

Using exactly one page (1-inch margins, 12-point Times New Roman font, single spaced, can include a small, simplified UML or other diagram(s) as a visual aid) describe your justification for the most important design decision that you made as a group in designing PaintShop. Note, this cannot be the decision to use masks – we imposed that design decision as a requirement of the project – this needs to be a design decision that you made yourself. This description will be most compelling if you can describe alternative designs that you considered within your group and then discarded. For example:

> *Tools are implemented in our design as follows: [describe in a few paragraphs, reference a UML diagram or other figure if this makes the explanation clearer].*
>
> *This design is the result of several hours of discussion and diagramming. During that process our team also considered two main alternatives designs.*
>
> *Alternative #1 was [describe in a paragraph].*
>
> *Alternative #2 was [describe in a paragraph].*
>
> *The main advantage of our final design as compared to Alternative #1 is [a few sentences]. Another advantage is [a few sentences]. Compared to Alternative #2, we believe our final design is better because [a few more sentences].*

Page 3 of the Group Design Document:

The same thing as Page 2, but for the second most important design decision. Keep in mind that the "most important" design decision might actually be the "most controversial" design decision. Think about what your group spent the most time discussing and describe the technical details of that discussion. We want to see that you put some serious effort into this design and that you learned something from it!

3. High-Level Grading Rubric

As you plan how to allocate tasks within your group, please keep in mind the following grading rubric that we will use. We include it here specifically to emphasize the importance of the design aspects of this project. It is possible to implement PaintShop through a brute force approach to programming that doesn't make good use of abstract data types or plan for future change, which you should know is coming since we have two more iterations of the project to complete in this course ☺. For the CSci-3081W course, we're not interested in the brute force approach, even if the implementation works perfectly. We want to see a thoughtful design that works not just now but also for the future. This is reflected in the way we will assess the work:

33% - Quality of the "most important design decisions" described in the Group Design Document.

33% - Quality of the design decisions made as evaluated by inspecting the actual source code.

33% - Quality of the implementation – meets the requirements for PaintShop. Note that in order to receive a perfect score for implementation, your Special tool needs to do something interesting. You will receive some credit, but not full credit, for a Special tool that is just a slight change to one of the other tools. Also, to receive a perfect score, your code must compile on the CSE Labs machines on the first try after we unzip your project source code and run make.

Total: 100%