Cairo University
Faculty of Engineering
Computer Engineering Department
Programming Techniques Course

Programming Techniques Project Requirements

Paint for Kids

Introduction

A fancy colorful application is an effective way to teach kids some computer skills. Educational games are another enjoyable way for kids teaching. In this project (Paint for Kids) we are going to build a simple application that enables kids draw fancy shapes and also play some simple games with those shapes. Your application should help a kid draw a number of figures, fill them with different colors, save and load a graph, and so on. The application should provide a game playing mode to teach kids how to differentiate between figures' types and colors.

Your Task:

You are required to write a C++ code for Paint for Kids application. Delivering a working project is NOT enough. You must use ***object-oriented programming*** to implement this application and respect the **responsibilities** of each class as specified in the document. See the evaluation criteria section at the end of the document for more information.

Project Schedule

Project Phase	***Deliverables***
Phase 1 [25%]	**Input-Output Classes**
Phase 2 [75%]	**Final Project Delivery**

The project code must be totally yours. The penalty of cheating from any other source (internet, other teams, etc.) is taking ZERO in the project grade. A plagiarism check will be applied to all teams' codes to detect any cheating cases.

Main Operations

The application supports 2 modes: **draw mode** and **play mode**. Each mode contains 2 bars: **tool bar** that contains the main operations of the current mode and **status bar** that contains any messages the application will print to the kid.

The application should support the following operations (actions) in each mode. Each operation **must** have an **icon** in the tool bar. The kid should click on the operation icon from the tool bar to choose it **first**, then give it its input(s) if needed.

Note: *Any* ***percentage*** *written below next to any operation is its percentage from* ***phase 2 grade****. See the evaluation criteria of phase 2 for more details.*

Percentages are added according to the **difficulty** of the task, so to divide the project load **equally** on team members, each member should take actions that their percentages add up to about **25%** of phase 2 (if 4 students in the team).

The main operations supported by the application are:

[I] Draw Mode: [75%]

Note: See the "General Operation Constraints" section below to know the general constraints that must be applied in any operation.

1- **[7.5%] Add Figure:** adding a new figure to the list of figures drawn on screen. This includes:
- ❑ Adding a new **rectangle**, **square**, **triangle**, **hexagon**, or **circle** (the figures initially are unfilled)
- ❑ **Note:** Make all **hexagon** figures have the same size, so you need to take only a click on the center point from the kid. The same note is applied in **squares** too but not in rectangles, triangle, or circles.

2- **[7.5%] Select One:** selecting **one** of the figures.
- ❑ The kid first chooses the "**select one**" icon from the tool bar
- ❑ Then clicks inside the figure or on its border (***same for filled and unfilled figures***)
- ❑ If a figure is selected before and the user clicks on the "select one" icon again:
  - ❑ If the kid re-clicks on the selected figure, this will un-select it.
  - ❑ If the kid clicks on an empty area (not inside any figure), this will be ignored.
  - ❑ If the kid selects a figure while another one is selected, the previous one will be unselected and the newly-selected one will be selected (NO multiple selection is allowed in this action)
- ❑ When the kid selects a figure,
  - ❑ This figure should be highlighted (Assume that the highlight color is "**magenta**"; a color in the library)
  - ❑ All information about the selected figure should be printed on the status bar.
    For example, the application can print (depending on figure type):
    the figure ID, start and end points, width, height, ...etc.

3- **[2.5%] Change "Figure" Colors:** changing the drawing or filling color (2 icons) for the selected figure and any subsequent drawings. You have to select figure first (using the select icon) before clicking the icon of changing its drawing or filling color. Existing figures will NOT be changed but the selected figure and any subsequent figures will be drawn using the changed drawing and filling colors (until they're changed again and so on). The available colors are: **black**, **yellow**, **orange**, **red**, **green,** and **blue**.

4- **[2.5%] Delete Figure:** deleting the selected figure.

5- **[5%] Move Figure:** moving the selected figure.
- ❑ The user first selects the figure (using the select icon) then clicks the move icon.
- ❑ Move action needs one extra click from the kid to specify the move destination.

6- **[7.5%] Undo:** undoing the last performed operation. If you clicked undo 2 times consecutively, this will undo the last two operations, and so on. You can click on the undo icon maximum 5 times consecutively to undo the last 5 performed operations.

7- **[7.5%] Redo:** redoing the undone operation. The redo icon can be clicked maximum 5 times consecutively. The redo icon can only redo if it is clicked immediately after an undo/redo, but no redo is done if its previous operation was not undo/redo.
- ❑ **The operations allowed to be undone/redone are**:
  Add Figure, Delete Figure, Change Color, and Move.
  Any other operations done between these operations are just ignored in the sequence of undo/redo.
- ❑ An example scenario:
  - ➢ Add rectangle R1, then R2, then R3, then R4, then R5.
    → Now we have (R1,R2,R3,R4,R5)
  - ➢ Undo. → Now we have (R1,R2,R3,R4)
  - ➢ "Select One" operation.
  - ➢ Undo. → Now we have (R1,R2,R3), note that the select is ignored.
  - ➢ Redo. → Now we have (R1,R2,R3,R4)
  - ➢ Redo. → Now we have (R1,R2,R3,R4,R5)
  - ➢ Redo. → no action because no more operations to redo
  - ➢ Undo. → Now we have (R1,R2,R3,R4)
  - ➢ Undo. → Now we have (R1,R2,R3)
  - ➢ Redo. → Now we have (R1,R2,R3,R4)
  - ➢ Add R6. →Now we have (R1,R2,R3,R4,R6) (now last action is R6, and R5 is missed)
  - ➢ Redo. → no action because not after an undo/redo
  - ➢ Undo. → Now we have (R1,R2,R3,R4) → will undo the last action R6
  - ➢ Undo. → Now we have (R1,R2,R3)
  - ➢ Undo. → Now we have (R1,R2)
  - ➢ Redo. → Now we have (R1,R2,R3)

8- **[2.5%] Clear All**: this action will clear/reset all actions performed by the kid: all figures are deleted, all history of undo/redo or recording becomes empty, and all program variables are reset as if the program has just started.

9- **[10%] Start and Stop Recording**: there are two icons: one to start recording and one to stop. Recording will record the operations done by the kid with maximum size of 20 operations.
- ❑ The recording can start only at the beginning of the program or immediately after a clear all, otherwise, an error message should be displayed
- ❑ The recording can record all draw-mode operations, except the following operations (excluded from recording): start/stop recording, play recording, save graph, load graph, switch, and exit.
- ❑ After stopping the recording, the kid can continue on the drawn figures normally.

10- **[5%] Play Recording**: this will play the last stopped recording with 1 second sleep between each two operations. When the kid chooses this icon, this will first perform the "clear all" operation (without the need for clicking on its icon) and then re-perform the recorded operations. All program variables should be changed as if these recorded operations are done live by the kid.

11- [7.5%] **Save Graph**: saving the information of the drawn graph (all the figures) to a file (see "file format" section in phase 2). The application must ask the kid about the filename to create and save the graph in (overwrite if the file already exists).

12- [7.5%] **Load Graph:** loading a saved graph from a file and re-drawing it (see "file format").
- ❑ This operation re-creates the saved figures and re-draws them.
- ❑ The application must ask the kid about the filename to load from.
- ❑ After loading, the kid can edit the loaded graph and continue the application normally.
- ❑ If there is a graph already drawn on the drawing area and the load operation is chosen, the application should clear the drawing area (make any needed cleanups of the current drawn graph) then load the new one.

13- [2%] **Switch to Play Mode:** by loading the tool bar and the status bar of the play mode. The kid can switch to play mode any time even before saving.

14- [0.5%] **Exit:** exiting from the application.
- ❑ Perform any necessary cleanup (termination housekeeping) before exiting.

[II] Play Mode: [20%]

In this mode, the graph created in the draw mode (or loaded from a file) is used to play some simple kids' games. The operations supported by this mode are:

1- [15%] **Pick & Hide:** The kid is prompted to pick the figures of a specific property (color or type). When he picks (clicks on) a figure, it should disappear and the kid picks the next figure with the same property value. The application should print counters to count the number of correct and incorrect picks done by the kid. Finally, when the kid picks all the figures of the required property, a grade is displayed for him showing how many correct and incorrect picks he made.
The kid should be able to pick figures by **one** of the following properties:
- ❑ **Figure Type:** e.g., pick all rectangles, …etc.
- ❑ **Figure Fill Color:** e.g., pick all red figures, …etc.
- ❑ **Figure Type and Fill Color:** e.g., pick all blue triangles.

You may need a toolbar icon for each one of the picking properties (3 icons). When the kid chooses a property (e.g., figure type or color) the application should randomly choose a property value (rectangle, line, …etc. if the property is figure type) and prompt the kid to pick it. The application should be logical when it chooses a random property value, for example, do not ask the kid to pick rectangles and the drawn graph does not have rectangles.

Note:
At any time, the kid can restart the game by clicking its menu icon. When the kid does so:
- ❑ The original graph should be restored.
- ❑ All changes done in the current game should be discarded.
- ❑ Don't forget to make any needed cleanups.

2- [5%] **Switch to Draw Mode**: At any time, kid can switch back to the drawing mode.
- ❑ The original graph should be restored
- ❑ All changes done in the play mode should be discarded.
- ❑ Don't forget to make any needed cleanups.

The remaining 5% of phase 2 is left for code organization and styling. See "Phase 2 Evaluation Criteria" section of phase 2 document for more info. This 5% is shared between all students.

[III] [Bonus] Operations [10%]:

The following operations are bonus and you can get the full mark without supporting them and you will not receive more bonus if you implemented any additional feature other than the features mentioned here in the bonus part. Anything related to the bonus will be delivered in **Phase 2.**

1- [5%] Add Voice to your application:

- ❑ When the kid makes any action like adding a new figure (e.g., rectangle), the application should say a word indicating what happened (e.g., say: “Rectangle” in a funny child voice).
- ❑ The icon of this action enables or disables the voice in your application.
- ❑ To take the full mark of this action, you need to add voice to at least 5 actions.

2- [2.5%] Move Figure(s) by Dragging: to move the selected figure by dragging a point inside it.

3- [2.5%] Resize Figure(s) by Dragging: to resize the selected figure by dragging its corners.

General Operation Constraints

The following rules must be satisfied even if not mentioned in the operation’s description:

1. Any needed **cleanups** (freeing any allocated memory) must be done.
2. Many operations needs some figures to be selected first. If no figures are selected before choosing such operations, an **error message** is displayed in the status bar and the chosen operation will make no change.
3. All kid inputs (except filename) must be obtained through **mouse clicks** on icons **not keyboard** typing. This is better to avoid typing errors and checks and also because the application targets kids so typing should be minimized.

Program Interface

The application window in draw mode may look like the window in the following figure. The window is divided to **tool bar**, **drawing area** and **status bar**. The tool bar of any mode should contain icons for all the actions in this mode (***Note***: the tool bar in the figure below is not complete and you should extend it to include all actions of the current mode).

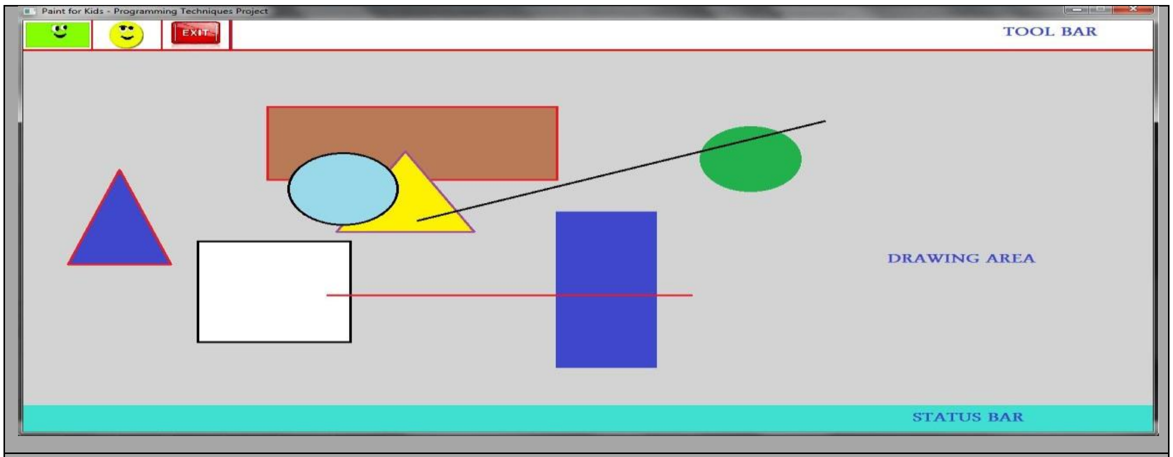

Figure 1 – Application screenshot

Main Classes

Because this is your first object-oriented application, you are given a ***code framework*** where we have ***partially*** written code of some of the project classes. For the graphical kid interface (GUI), we have integrated an open-source ***graphics library*** that you will use to easily handle GUI (e.g. drawing figures on the screen and reading the coordinates of mouse clicks …etc.).

You should **stick to** the **given design** (i.e., hierarchy of classes and the specified job of each class). We want you to **stick** to the given design because it is your **first OOP project** and we want to give you an example of a fairly-good designed code to work in it. In most of the projects of next years, you will be free to design your OOP classes the way you want.

NOTE: The application should be designed so that the types of figures and types of operations can be easily extended (*using inheritance*).

Below are the classes of phase 1.

Input Class:
ALL kid inputs must come through this class. If any other class needs to read any input, it must call a member function of the input class. You should add suitable member functions for different types of inputs.

Output Class:
This class is responsible for **ALL** GUI outputs. It is responsible for toolbar and status bar creation, figures drawing, and for messages printing to the kid. If any other class needs to make any output, it must call a member function of the output class. You should add suitable member functions for different types of outputs.

Notes: - No input or output is done through the console. All must be done through the GUI window.
- Input and Output classes are the **ONLY** classes that have direct access to **GUI library**.

Notes on The Project Graphics Library:

- ❑ The origin of the library's window **(0, 0)** is at the **upper left** corner of the window.
- ❑ The direction of **increasing** the **x coordinate** is to the **right**.
- ❑ The direction of **increasing** the **y coordinate** is **down**.
- ❑ The images extension that the library accepts is "**jpg**".

For **fast navigation** in the given code in **Visual Studio**, you may need the following:

- ❑ **F12 (go to definition):** to go to definition of functions (code body), variables, …etc.
- ❑ **"Ctrl" then "Minus":** to return to the previous location of the cursor.

Project Phase I

A partially implemented code frameworks for Phase 1 (and another for Phase 2) are given to you to complete them. You should gain the skill of how to continue on a partially implemented code.

1- Phase 1 (Input / Output Classes) [25% of total project grade]

In this phase, you will implement the **Input** and the **Output** classes as they do not depend on any other classes. The **Input** and **Output** classes should be **finalized** and ready to run and test. Any expected user interaction (input/output) that will be needed by phase 2, should be implemented in phase 1.

Input and Output Classes Code and Test Code

You are given a code for phase 1 (separate from the code of the whole project) that contains both the input and output classes partially implemented. Each team should complete such classes as follows:

1- ***Output Class***:

- ❑ **Draw the full tool bars**.
  Output class should create 2 FULL tool bars: one for the **draw mode** and one for the **play mode**. Each contains icons for every action in this mode.
  [Notes]:
  - ❑ Some operations need **more than one icon**, for example, "Pick and Hide" should contain 3 icons for the 3 types of picking properties (by type, by color, or both).
  - ❑ The toolbar of the draw mode must contain icons for the **colors** that may be used as drawing or fill color.
  - ❑ All icons must be added in phase 1.
- ❑ **A draw function for each figure type** (**rectangle**, **square**, **triangle**, **hexagon**, and **circle**). These functions must be able to draw the figures: either filled or not filled, with all supported colors for borders or filling, and draw them either normal or highlighted.
- ❑ Add any other needed member data or functions

2- Input Class:

- ❑ Complete the function **Input::GetUserAction(…)** where the input class should detect all possible actions of any of the 2 modes according to the coordinates clicked by the kid.
- ❑ The input class should also recognize the color selected by the kid from the color **palette.**
- ❑ Add any other needed member data or functions

3- ***Test Code:*** this is not part of the input or the output classes; it is just a test code (the **main**).

- ❑ Complete the code given in **TestCode.cpp** file to test both Input and Output classes.
- ❑ Do NOT re-write the main function from scratch, just complete the required parts.

Phase 1 Deliverables*:*

One .zip file containing:

- ❑ **IDs.txt** (team number, member names, IDs, email).
- ❑ **Phase 1 code** that has Input and Output classes and test program completed.
- ❑ **Work load distribution document** that contains what each member has done in phase 1.

Note: making all members work in all functions is NOT a successful distribution, and you will lose marks for that. Divide the functions among team members and then integrate.