CS 210 – Intro to Programming

PEX 3

2D Graphics Manipulation

**Due Date/Time**: This PEX has multiple due dates/times.

Gate1 - GUI and Matrix Multiplication: Demonstrate by end of class Lesson 26.

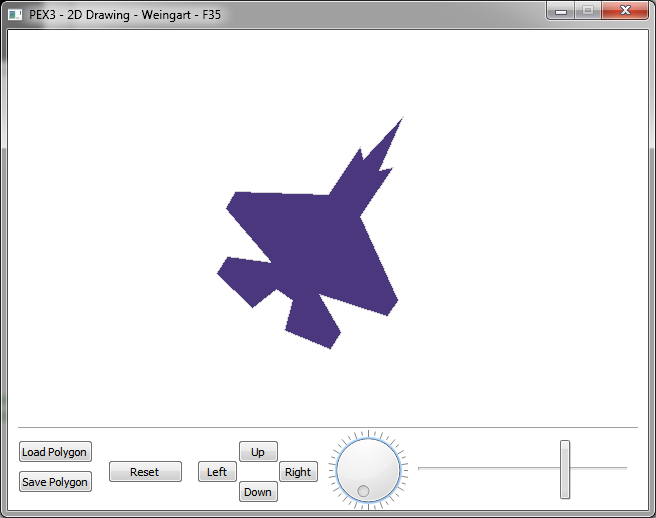
Gate2 - Load Polygon from File: Demonstrate by end of class Lesson 28.

Final - 2D Graphics Manipulation: Submit by 23:00 on Lesson 29 (5 Nov M-day, 6 Nov T-day).

|  |
| --- |
| Help Policy: **AUTHORIZED RESOURCES:** Any, except another cadet’s program.  **NOTE:**   * Never copy another person’s work and submit it as your own. * Do not jointly create a program. * You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook). * **DFCS will recommend a course grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.**  Documentation Policy:  * You must document all help received from any source other than your instructor. * The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance. * If no help was received on this assignment, the documentation statement must state “NONE.” * If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why. * **Vague documentation statements must be corrected before the assignment will be graded and will result in a 5% deduction on the assignment.**  Turn-in Policies:  * On-time turn-in is at the specific time listed above. * Late penalties accrue at a rate of 25% per 24-hour period past the on-time turn-in date and time. The late penalty is a cap on the maximum grade that may be awarded for late work. * There is no early turn-in bonus or extra credit for this assignment. |

**PEX Objectives:**

* Implement matrix manipulation techniques.
* Be able to read and write data to a file.
* Learn how matrix transformations can be used for computer graphics.
* Demonstrate good software design principles with a focus on functional decomposition.
* Demonstrate use of OO programming techniques.
* Practice robust coding techniques.
* Demonstrate use of graphics user interface components and dialog boxes.



**Background and Requirements[[1]](#footnote-1):**

Interesting graphical manipulations can be accomplished with matrix multiplication. You will be creating a program which uses matrix multiplication to manipulate and display a filled polygon. You are NOT allowed to use any libraries to accomplish the matrix multiplication; you must write your own method to accomplish this task. The method must be named “multiply” and it must be defined in your polygon class.

Your program will:

* allow the user to load and save a polygon (as a series of XY coordinates stored in a text file)
* allow the user to perform a series of graphical manipulations on a polygon – scale, rotate, and translate (move)
* utilized a graphical user interface developed in Qt Designer
* use file dialog boxes for the specification of the file to load/save
* use OO programming techniques to fulfill the requirements of this PEX

**Gate Checks (due in-class Lesson 27)**

For this assignment we require you to accomplish two gate checks. For each of these gate checks you need to have accomplished a particular set of subtasks related to the PEX requirements.

Gate Check-1: Have a fully drawn GUI that looks like the diagram above. None of the underlying code for the controls need to be implemented for this gate check (e.g., the button does not need to do anything other than be on the screen). Further, this gate check requires you to have implemented a correctly functioning Matrix multiplication operation.

Gate Check-2: Your GUI must be able to display a filled polygon, such as that stored in f35.txt in the drawing panel.

**Implementation Suggestions**:

* Read through this document and the provided Python code so you understand what is expected and what is provided.
* This is a non-trivial program that is best approached using functional decomposition. The instructor’s solution used the following methods in the PolygonApp class (PEX3\_PolygonApp.py):
  + \_\_init\_\_() – creates the default image (square) and connects GUI components to the application's methods.
  + load\_poly() – loads a polygon from a file. *(required for gate 2 check)*
  + save\_poly() – saves a polygon to a file.
  + reset() – restores a polygon to it’s initially loaded state.
  + paintEvent() – method called by Python runtime that draws the current polygon.
  + There will also be a series of functions that perform the user selected graphical manipulation on the polygon. (e.g. scaleUP(), scaleDown(), rotate(),etc).
* Additionally, the instructor’s solution used the following methods in the Polygon class (Polygon.py):
  + multiply(self, other) - performs matrix multiplication of two matrices, defined as nested lists, and returns the resulting product matrix as a nested list (this is non-scalar multiplication!) Also be sure to not modify the original matrix. This method should return a new matrix.
    - The following links may provide some background on matrix multiplication
      * <https://www.khanacademy.org/math/precalculus/precalc-matrices/matrix_multiplication/v/multiplying-a-matrix-by-a-matrix>
      * <http://www.mathwarehouse.com/algebra/matrix/multiply-matrix.php>
  + transform(self, operation)- performs given transform on a matrix, for example a call to this function to perform a scale-up operation might look like, current\_object.transform(“scaleup”).
  + translate\_to\_drawable\_polygon(self)- converts a polygon matrix that is defined as nested lists into a QtGUI.QPolygon to allow the polygon to be drawn with a QtGui.Painter object’s drawPolygon() method.
* It is recommended that you implement this program in an incremental manner, writing and testing at most one method (or part of a method) before going on to the next part.
  + You can do this in the **top-down method** starting with the major methods and then having those call methods that currently do nothing (using the *pass* command) but possibly return a set value (these are called “stubs”). Once that works, implement one of the stubbed methods; creating stubs for other methods that it will need to call.
  + Another approach is the **bottom-up method** where you start with methods such as multiply() and use a few lines of code in the main method to test them.
  + Some prefer the **sandwich method** which combines the top-down approach for some methods with a bottoms-up approach for other. They meet in the middle; “sandwich”ing the results. I would suggest that using this approach as the first gate explicitly tests your multiply method and your GUI.
* You will create transformed polygons by multiplication of the appropriate transformation matrix times the matrix describing the polygon for the current shape. The transformation matrices are provided in the Polygon.py template. Be sure you perform the multiplication with the arguments in the correct order (*ResultMatrix =* *TransformMatrix \* CurrentPolygonMatrix)*.
* Ensure your code is properly commented.
* Sample input/output files and templates are provided for your use (look at them!).
  + Saved polygons are f35.txt, note.txt, square.txt
  + The two provided templates are Polygon.py, PEX3\_PolygonApp.py
* Take a good look at the python template files and the transformation matrices provided. The three rows of your polygon matrix should be defined as follows: the first row will be the x-coordinates for each points in the polygon, the second row is the y-coordinates for each of the point in the polygon, the third row is all 1’s. The 3rd row of ones is used only when performing the matrix math and is not used when displaying the polygon. A nested list describing a square with points (-50,-50), (50,-50), (50,50) and (50,-50) is shown below.

square = [[ -50, 50, 50, -50],[-50, -50, 50 , 50],[1, 1, 1, 1]]

* Document ALL HELP received other than that from your instructor, the course text, lesson notes/slides, and course example programs.
  + Provide your DOCUMENTATION STATEMENT in a separate area of the header comment block in your PEX3\_PolygonApp.py file. Be sure to include ALL of YOUR DOCUMENTATION.
* Submit your \*.ui file, all \*.py files, and any object (i.e., image) files you wish your instructor to use via the link on the course web site.
  + If for some reason, this submission link does NOT work, simply email the required file(s) to your instructor.

**CS 210 – PEX 3 Grade Sheet Name:**

**Section:**

Points

Criteria Earned Available

|  |  |  |  |
| --- | --- | --- | --- |
| Gate Checks (10 points) | |  |  |
| Gate 1 – GUI loads and Polygon.multiply() works (5 points – all or none). | |  | **5** |
| Gate 2 – GUI loads and displays a Polygon (5 points – all or none). | |  | **5** |
| Documentation (16 points) | |  | **0** |
| Standard program header comment block | |  | **4** |
| Standard function header comment blocks | |  | **6** |
| Appropriate commenting throughout | |  | **6** |
| Readability and Maintainability (26 points) | |  | **0** |
| Program formatting, meaningful/consistent identifier names, appropriate use of named constants, etc. | |  | **6** |
| Appropriate functional decomposition | |  | **12** |
| Adheres to generally accepted programming practices including arranging the source file to use a main() function, avoids use of global variables, etc. | |  | **8** |
| Functionality (48 points) | |  | **0** |
| Well-designed, resizable, graphical user interface. | |  | **8** |
| Reads polygon file and stores data in an appropriate data structure  (including appropriate handling of Cancel/Close on file dialog). | |  | **8** |
| Saves polygon file correctly (including appropriate handling of Cancel/Close on file dialog). | |  | **8** |
| Polygon is correctly drawn on screen and updates appropriately when reset. | |  | **8** |
| Transformations work correctly. | |  | **16** |
| **Subtotal:** | |  | **100** |
| **Adjustments** | **Vague/Missing Documentation:** |  | **− 5** |
| **Submission Requirements Not Followed:** |  | **− 5** |
| **Late Penalties:** |  | **25/50/75%** |
| **Total w/adjustments:** |  |  |

Comments:

1. The initial idea for this Pex was provided by Dr. Steve Hadfield [↑](#footnote-ref-1)