

PA2 – Math

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: Yes No

Name:

Date:

Submission Details

Final **Changelist** number:

Verified build: Yes No

Number Tests Passed:

Required Configurations:

Discussion (What did you learn):

Verify Builds

- Follow the Piazza procedure on submission
 - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
 - No – Generated files
 - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
 - Anything that is generated by the compiler should not be included
 - No – Generated directories
 - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
 - *.sln, *.cpp, *.h
 - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
 - As soon as you get something working, submit to perforce
 - Have reasonable check-in comments
 - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
 - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
 - Warning level ALL ...
 - NO Warnings or ERRORS
 - Your code should be squeaky clean.
 - Code needs to work "as-is".
 - No modifications to files or deleting files necessary to compile or run.
 - All your code must compile from perforce with no modifications.
 - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason...
 - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
 - No automatic containers or arrays
 - You need to do this the old fashion way - **YOU EARNED IT**

Leave Project Settings

- Do NOT change the project or warning level
 - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
 - No Lambdas, Autos, templates, etc...
 - No Boost
- NO Streams
 - Used fopen, fread, fwrite...
- No code in MACROS
 - Code needs to be in cpp files to see and debug it easy
- **Exception:**
 - implicit problem needs templates

Leaking Memory

- If the program leaks memory
 - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
 - It is responsible for its deletion
- Any **MEMORY** dynamically allocated that isn't freed up is **LEAKING**
 - Leaking is **HORRIBLE**, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
 - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
 - All files must be active to get credit.
 - Better to lose points for unit tests than to disable and lose all points

No Adding files to this project

- This project will work "as-is" do not add files...
- Grading system will overwrite project settings and will ignore any student's added files and will returned program to the original state

UnitTestFixture file (if provided) needs to be set by user

- Grading will be on the UnitTestFixture settings
 - Please explicitly set which tests you want graded... no regrading if set incorrectly

Due Dates

- See Piazza for due date and time
- Submit program performance in your student directory assignment supplied.
- Fill out your this **Submission Report** and commit to performance
 - **ONLY** use Adobe Reader to fill out form, all others will be rejected.
 - Fill out the form and discussion for full credit.

Goals

- Create the Vector and Matrix classes for the math engine
 - Use a Unit Test environment
- Reverse engineering
 - Debug and test your programming skills

Assignments

1. Create the Math program in C++, for Visual Studio 2022

- a. Document the code
- b. Code should be Warning Level Wall free
 - It doesn't compile -> 0 for grade
- c. Compiles in all configurations
 - Debug x86
 - Release x86

2. Create a stand-alone Math Library for Game Engine

- a. Write a Vector Classes
 - Vec3
 - Vec4
- b. Write a Matrix Classes
 - Mat3
 - Mat4
- c. Write a wrapper Trig Class
- d. Proxies for Vector length
 - Vec3Proxy
 - Vec4Proxy
- e. Create any necessary classes and code to complete the library
- f. Document the code
- g. Code should be warning free

3. Implement the functions and api

- a. You have freedom to implement the code in any way or style you desire
- b. Keep it clean and neat, with reasonable documentation
 - All classes have suggested API that correspond to the Unit test
- c. You need to keep use the existing Vector and Matrix data definitions
 - Proxy definitions can change
 - Idea avoid unnecessary square root functions with the proxies
- d. Data must remain in the private section
- e. Trig class is just a wrapper class of static functions
 - Anywhere you need a trig function.. use that wrapper class
- f. No SIMD
 - This can be added between quarters if desired

4. Unit test framework is provided

- a. All tests should run for full credit
- b. You cannot modify the tests given

5. Const correctness and prototypes will be reviewed

- a. Try to use references
 - Suggested api is const and reference correct
- b. Try to use const as much as possible

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run **ALL** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
 - Is all the code there and compiles “as-is”?
 - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do this assignment by iterating and slowly growing your project
 - Use the test code for your use cases
- This is a fairly easy assignment, but tedious.
 - Look at the lecture slides for the math.
 - There are websites with additional math or ask to the forum

- The hardest part might be figuring out the API
 - Reverse engineer the API from the test code.
 - consts might give you troubles, so be aware of the different consts
- Your tests will not run at first since you haven't implement code yet.
 - Focus on the first tests, getting the constructor and bracket operator working.
- I would comment in one test group, then focus on the one test at the time
 - Baby steps
 - Incremental development!
- You are learning 2 things, test-driven development and reverse engineering.
 - Both are very useful, but require a little thinking.
- Please
 - Draw diagrams to help you understand