DigiPen Institute of Technology

CSD1130

Game Implementation techniques

Assignment 4 – Part 1

2D Vector Library – 2D Matrix Library

Due date:
Section A: Thursday, March 16th, 2023, at 2:00pm
Section B: Thursday, March 16th, 2023, at 4:30pm

Topics
The assignment will cover the following topics:

1. Creating a 2D vector library
2. Creating a 2D matrix library

Goal
The goal of this assignment is to implement 2D vector and matrix libraries, which will be later used by the 2nd part of this assignment, which is implementing a cage simulation.

Submission Guidelines – Visual Studio [Using Microsoft C++ Compiler]

- Submit at Moodle entry:
  "**csd1130_Assignment_4_Part1_Submission_VisualStudio**"
- To submit your programming assignment, organize a folder consisting of Vector2D.cpp and Matrix3x3.cpp **ONLY**. Name this folder using the following convention:

  <class>_<section>_<student login name>_<assignment#>_<part#>

  For example, if your login is ***foo.boo*** and assignment 4 part 1 is being submitted, your folder would be named **csd1130_a_foo.boo_4_1**
- Your folder must not contain any extra file.
- Zip this folder and name the resulting file using the following convention:

  <class>_<section>_<student login name>_<assignment#>_<part#>.zip

  For example, if your login is ***foo.boo*** and you are submitting assignment 4 part 1, your zipped file would be named as: **csd1130_a_foo.boo_4_1.zip**
- Next, upload your zip file after logging into the course web page using the link https://distance3.sg.digipen.edu.

- Finally, perform a sanity check to determine if your programming submission follows the guidelines by downloading the previously uploaded zip file, unzipping it, then compiling, linking, and executing your submission.

Using the right Compiler and MSVS setup

- The project must be tested in **RELEASE and DEBUG** modes with **warning level 4**, under **x64** platform. Under project settings the **SDK Version**: must be set to *10.0 (latest installed version)* and the **Platform Toolset** set to *Visual Studio 2019 (v142)*. It must generate 0 warnings and 0 errors. This can be verified on a PC located at **Pascal lab**.

- Please validate the previous statement before your submission!

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 4 – Part 1
2D Vector Library – 2D Matrix Library

Submission Guidelines – VPL [Using g++ Compiler under]

- Submit under this VPL entry, after you finish and submit your work under the Visual Studio entry, above!
- Submit at Moodle entry: "**csd1130_Assignment_4_Part1_Submission_VPL**"
- To submit your programming assignment:
    - Upload your Vector2D.cpp and Matrix3x3.cpp code file, done in the visual studio project.
    - Run & Evaluate
        - Correct evaluation will give the following result:
            - **Summary of tests**
            - [23 test run/ 23 test passed]

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 4 – Part 1
2D Vector Library – 2D Matrix Library

Description

- ✓ This project must be implemented as a Win32 **Console** Application (*start with an empty project*)
- ✓ When you create the visual studio project, put all the code files together, near the ".vcxproj' file!
- ✓ Language: C/C++ and your code should be commented!
- ✓ A test driver is provided to test your implemented functions. (**Do not modify this file!**)
- ✓ Two header files will be provided. (**Do not modify these files!**)
  - They will include the functions' declarations for the vector and matrix libraries.
  - They will also include the vector and matrix structure which will be used by these functions.
- ✓ You must implement a 2D vector library.
  - The provided header file for this library is "Vector2D.h"
  - This library is responsible for all 2D vectors functionalities.
  - The 2D vector header file contains the vector's structure. This structure will be used by all vector related functions.
  - Implementation:
    - ➢ Implement the functions of the 2D vector library in "Vector2D.cpp"
    - ➢ The 2D vector structure (provided in the Vector2D.h header file) is

```
typedef union Vector2D
{
	struct
	{
		float x, y;
	};

	float m[2];

	// Constructors
	Vector2D() {};
	Vector2D(float _x, float _y);

	// Assignment operators
```

```
        Vector2D& operator += (const Vector2D &rhs);
        Vector2D& operator -= (const Vector2D &rhs);
        Vector2D& operator *= (float rhs);
        Vector2D& operator /= (float rhs);

        // Unary operators
        Vector2D operator -() const;

    } Vector2D, Vec2, Point2D, Pt2;
```

- The 2D vector functions that you must implement on top of the overloaded operators and the conversion constructor (default constructor not implemented) in the Vector2D structure are:
  - ➢ `Vector2D operator + (const Vector2D &lhs, const Vector2D &rhs);`
  - ➢ `Vector2D operator - (const Vector2D &lhs, const Vector2D &rhs);`
  - ➢ `Vector2D operator * (const Vector2D &lhs, float rhs);`
  - ➢ `Vector2D operator * (float lhs, const Vector2D &rhs);`
  - ➢ `Vector2D operator / (const Vector2D &lhs, float rhs);`
  - ➢ `void Vector2DNormalize(Vector2D &pResult, const Vector2D &pVec0);`
  - ➢ `float Vector2DLength(const Vector2D &pVec0);`
  - ➢ `float Vector2DSquareLength(const Vector2D &pVec0);`
  - ➢ `float Vector2DDistance(const Vector2D &pVec0, const Vector2D &pVec1);`
  - ➢ `float Vector2DSquareDistance(const Vector2D &pVec0, const Vector2D &pVec1);`
  - ➢ `float Vector2DDotProduct(const Vector2D &pVec0, const Vector2D &pVec1);`
  - ➢ `float Vector2DCrossProductMag(const Vector2D &pVec0, const Vector2D &pVec1);`

- A description of each function is found in the provided “Vector2D.h” header file.

- ✓ You must implement a 2D matrix library.
  - The provided header file for this library is “Matrix3x3.h”
  - This library is responsible for all 2D matrix functionalities.
  - The 2D matrix header file contains the matrix' structure. This structure will be used by all matrix related functions.
  - Implementation:
    - ➢ Implement the functions of the 2D matrix library in “Matrix3x3.cpp”
    - ➢ The 2D matrix structure (provided in the in Matrix3x.h header file) is:

```
 typedef union Matrix3x3
{
  struct
  {
  float m00, m01, m02;
  float m10, m11, m12;
  float m20, m21, m22;
  };

  float m[9];
  float m2[3][3]; // to use in Assignment’s part 2

  Matrix3x3() {}
  Matrix3x3(const float *pArr);
  Matrix3x3(float _00, float _01, float _02,
      float _10, float _11, float _12,
      float _20, float _21, float _22);
  Matrix3x3& operator = (const Matrix3x3 &rhs);

  // Assignment operators
  Matrix3x3& operator *= (const Matrix3x3 &rhs);

} Matrix3x3, Mtx33;
```

- The 2D matrix functions that you must implement on top of the overloaded operators and the conversion constructors (default constructor not implemented) in the Matrix3x3 structure are:
  - `Matrix3x3 operator * (const Matrix3x3 &lhs, const Matrix3x3 &rhs);`
  - `Vector2D  operator * (const Matrix3x3 &pMtx, const Vector2D &rhs);`
  - `void Mtx33Identity(Mtx33 &pResult);`
  - `void Mtx33Translate(Mtx33 &pResult, float x, float y);`
  - `void Mtx33Scale(Mtx33 &pResult, float x, float y);`
  - `void Mtx33RotRad(Mtx33 &pResult, float angle);`
  - `void Mtx33RotDeg(Mtx33 &pResult, float angle);`
  - `void Mtx33Transpose(Mtx33 &pResult, const Mtx33 &pMtx);`
  - `void Mtx33Inverse(Matrix3x3 *pResult, float *determinant, const Matrix3x3 &pMtx);`

- Note that you should implement the "column-matrix" approach. For example, a translation matrix should hold the translation values in its last column.

- A description of each function is found in the provided “Matrix3x3.h” header file.

Finally, each ".cpp" file in your homework should include the following header:

```
/* Start Header *****************************************************************/
/*!
\file         <put file name here> (e.g. main.cpp)
\author       <provide your name, student login, and student id>
              (e.g. DigiPen Singapore, dSingapore, 60001906)
\par          <provide your email address> (e.g. email: digipen\@digipen.edu)
\date         <date on which you created this file> (e.g. Jan 01, 20xx)
\brief        <provide a brief description of the file content>

Copyright (C) 20xx DigiPen Institute of Technology.
Reproduction or disclosure of this file or its contents
without the prior written consent of DigiPen Institute of
Technology is prohibited.
 */
/* End Header *******************************************************************/
```

Evaluation

Here are the most common reasons assignments are marked down:

- Project does not build.
- Project does not build without warnings.
- One or more items in the "Requirements" section was not satisfied.
- A fundamental concept was not understood.
- Code is sloppy and hard to read (e.g. indentation is not consistent, no comments, etc…)
- Your solution is difficult (or impossible) for someone reading the code to understand due to lack of comments, poor variable/method names, poor solution structure, etc…
- Project assignment was turned in late.

Grading Algorithm

This project will be graded according to the following rules. Any rule that is missing, points will be deducted from the project's grade:

- Compile errors or does not compile for some reasons.
- Compile Warnings.
- Program runtime crashes.
- Output not matching a "**Pass**":
    - Every function that prints "**Fail**".
- If having Memory leaks!

Notes for Moodle submissions:

- "Visual Studio" submission is **NOT** counted for A4_P1 grading!
- "VPL" submission is counted as the full score for A4_P1 grading.

Good Luck!