Fall 2013
CS 529 | Fundamentals of Game Development
Project 1 | Asteroids

Files (submit folder) due

- **Part 2** – Tuesday, September 24th, 2013
- 11.55pm

Topics

The assignment will cover the following topics

1. Implement an "Asteroids" game, including:
    a. Building a 2D matrix library
    b. Building a 2D collision library
    c. Building a 2D vector library
    d. Ship movement based on acceleration, velocity (and velocity cap)
    e. Asteroids movement based on velocity
    f. Bullet spawning and movement based on velocity
    g. Collision checking
    h. Homing missile

Goal

The goal of this assignment is to implement a 2D asteroids game, which will include matrix, vector and collision libraries, in addition to the implementation of "physics movement" which will be used to update the positions of the game object instances.

Assignment Submission

- Compress (.zip) the solution folder (Delete the debug/release folders and the .ncb file first), and submit it on distance.digipen.edu.

Check the course syllabus regarding the naming and submission convention.

Description

I. Language: C

II. A start-up application will be provided.

III. A library will be provided (For part 2), which includes several hardware related functions like initializing/updating and freeing the graphics and input engines.
   a. Library name: "Alpha_Engine.lib"/ "Alpha_Engine.dll"
   b. The header files of the "Alpha_Engine.lib"/ "Alpha_Engine.dll" library are included in the solution folder.

IV. The project is divided into 2 parts
   a. Part 1: Implement the vector, transformation and static intersection libraries
   b. Part 2: Implement the Asteroids game

Part 1

I. Implement the 2D vector library
   a. Function declarations are found in Vector2D.h
   b. Implement the functions in Vector2D.c
   c. Detailed explanations are found in the .h file

II. Implement the transformation library
   a. Function declarations are found in Matrix2D.h
   b. Implement the functions in Matrix2D.c
   c. Detailed explanations are found in the .h file

III. Implement the static intersection library
   a. Function declarations are found in Math2D.h
   b. Implement the function in Math2D.c
   c. Detailed explanations are found in the .h file

Part 2

I. Implement the Asteroids game

II. No additional files should be created nor added to the project.

III. Copy your matrix, vector and math .c and .h files to the solution folder and add them to the project.

IV. GameStateAsteroids.h
   a. No changes should be made to this file.

V. GameStateList.h
   a. No changes should be made to this file.

VI. GameStateMgr.h
   a. No changes should be made to this file.

VII. GameStateMgr.c

a. Implement the "GS_ASTEROIDS" case in the "GameStateMgrUpdate" function.

VIII. GameStateAsteroids.c

a. **Make sure to replace all the vector and matrix variables and functionalities by your own. The Alpha Engine library I'll use to test your project won't have any Vector/Matrix/Collision functionalities!**
   - Example: Replace AEVec2 by Vector2D, AEMtx33 by Matrix2D.

b. In the "GameStateAstreoidsLoad" function:
   - Create the game objects (shapes): Ships, Bullet, Asteroid and Missle
   - Remember to create **normalized** shapes, which means all the vertices' coordinates should be in the [-0.5;0.5] range. Use the object instances' scale values to resize the shape.
   - Call "AEGfxTriStart" to inform the graphics manager that you are about the start sending triangles.
   - Call "AEGfxTriAdd" to add 1 triangle.
     - A triangle is formed by 3 **counter clockwise** vertices (points).
     - Create all the points between (-0.5, -0.5) and (0.5, 0.5), and use the object instance's scale to change the size.
     - Each point can have its own color.
     - The color format is: ARGB, where each 2 digits represent the value of the Alpha, Red, Green and Blue respectively. Note that alpha blending (Transparency) is not implemented.
     - Each point can have its own texture coordinate (set them to 0.0 in case you're not applying a texture).
   - An object (Shape) can have multiple triangles!
   - Call "AEGfxTriEnd" to inform the graphics manager that you are done creating a mesh, which will return a pointer to the mesh you just created.

c. In the "GameStateAsteroidsInit" function:
   - Create 3 initial asteroids instances outside the viewport, and make sure they're moving towards it.
   - Each asteroid should have a different size.

d. In the "GameStateAsteroidsUpdate" function:
   - Update the ship's acceleration/velocity/orientation according to user input.
     - AEInputCheckCurr: Checks pressed keys
     - AEInputCheckTriggered: Checks triggered keys
     - Make sure to cap the ship's velocity (Emulating friction)
   - Create a bullet when the space key is triggered
   - Create a homing missile when the 'm' key is triggered
   - Update game object instances' positions according to their velocities.

- Update specific game object instances' behavior.
  - Wrapping the asteroids and the ship around the world is required.
  - Removing the bullets when they go out of bounds is required.

- Check for collision between active game objects.
  - Pick an appropriate collision data for each game objects, as long as it makes sense.
  - Example that makes sense: Using a point as the bullet's collision data.
  - Example that doesn't make sense: Using a point as the asteroid or ship's collision data.
  - **Take the object instance's scale value into consideration when checking for collision.**

- Calculate the transformation matrix of each active object instance.
  - Remember that the order of matrix concatenation is important!
  - Order of matrix concatenation: Scale, then rotation, then translation.

e. In the "GameStateAsteroidsDraw" function, do the following for each game object instance:
   - Set the transformation matrix of each game object instance using the "AEGfxSetTransform".
   - Draw the object instance using the "AEGfxTriDraw" function.
     - Typecast your Mtx33 transform matrix to AEMtx33 when passing it to the AEGfxTriDraw function.

f. In the "GameStateAsteroidsFree" function:
   - Kill each **active** game object instance using the "gameObjInstDestroy" function.

g. In the "GameStateAsteroidsUnload" function:
   - Free each game object (shape) using the "AEGfxTriFree" function.

Finally, each ".c" and ".h" file in your homework should include the following header:

```
/* Start Header -------------------------------------------------------

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.

File Name:      <put file name here>
Purpose:        <explain the contents of this file>
Language:       <specify language and compiler>
Platform:       <specify compiler version, hardware requirements, operating
systems>
```

```
Project:        <specify student login, class, and assignment. For example:
                if foo.boo is in class CS 529 and this file is a part of
                assignment 3, then write: CS529_fooboo_3>
Author:          <provide your name, student login, and student id>
Creation date:   <date on which you created this file>

- End Header ----------------------------------------------------*/
```