# Due Date

This assignment must be completed and submitted via Moodle before end-of-day on Friday during Week 12 (Spring Semester) or Week 9 (Summer Semester).

# Objectives

The objectives for this project are four-fold:

* Implement a new game state (“Omega”)
* Implement separate colliders for circles and lines
* Implement Circle-Line collision detection and reflection
* Implement a score system for use by two different game states

# Description

For this project, you have been provided with a set of header files (.h) that specify the interface for four new modules. You are responsible for creating the associated source files (.c) and implementing the functionality, as outlined in the header files and the lecture notes.

The game state (Asteroids) created in Projects 4 and 5 will be duplicated in order to implement a new game state in Project 6. All game states created during the previous projects should remain in the game and be accessible from the Asteroids and Omega game states.

# Suggested Steps

1. Migrate your source code (.c files) from Project 5 to Project 6
2. Commit the project to your version control solution (SVN, Git, etc.)
3. Implement any stub functions required to get the project to compile. Use temporary return values (0 or NULL) and UNREFERENCED\_PARAMETER() macros to solve any compiler warnings
4. Commit the project to your version control solution (SVN, Git, etc.)
5. Refactor the scoring and HUD Text code into a new ScoreSystem module
   * Verify that the Asteroids level still works properly
6. Implement the ColliderCircle functionality
   * Verify that the Asteroids level still works properly
7. Create a new Omega game state by duplicating the Asteroids level and updating GameStateTable.c/.h
   * Verify that the Omega level works exactly like the Asteroids level
8. Add the Arena object to the Omega level
   * Verify that the inner rectangle appears correctly
9. Switch the HUDText objects from the “Asteroids” versions to the “Omega” versions.
   * Verify that the HUD text now appears within the inner region of the Omega level
10. Implement the ColliderLine functionality and update GameStateOmega to use this new functionality
    * Verify that objects correctly bounce off the arena walls

# Files

NOTE: You may not change the public interface of the header files (.h) that were provided in Projects 2 through 6, except as expressly directed in the instructions below. Should you modify these header files in any way, exercise extreme caution, as adding, removing, or modifying the public interface will result in a penalty to your project grade.

NOTE: The Animation, ColliderCircle, ColliderLine, Physics, Sprite, SpriteSource, SpriteSourceManager, Transform, GameObject and GameObjectManager and MeshManager structures must all be declared in their associated .c files, not the .h files. Exposing the internal implementation of these modules by declaring the structures in the .h files will result in a penalty to your project grade.

The Behavior and Collider structures are declared publicly in the .h file, as they will be used to implement pseudo-inheritance in this project.

ScoreSystem.h

* This header file declares the public interface for managing and displaying the score, high score, and wave count variables used by the Asteroids and Omega game states

ScoreSystem.c

* The majority of the code in this module will be migrated over from the GameStateAsteroids module. The existing code in GameStateAsteroids will be replaced with calls to functions within this module
* Migrate the private variables for score, high score, and wave count from GameStateAsteroids
* Implement the public functions, as described in the header file, by migrating the existing code from GameStateAsteroids

BehaviorAsteroid.c

* BehaviorAsteroidCollisionHandler
  + Replace the call to GameStateAsteroidsIncreaseScore() with a call to ScoreSystemIncreaseScore()
* BehaviorAsteroidSetPosition:
  + The asteroids should spawn in the four corners of the screen. If the asteroids remain “stuck” in the corners after they are spawned, then you likely have one or more flaws in the ColliderLine module or in the ordering of the function calls within the main game loop

BehaviorHudText.h

* The following changes have been made to this header file:
  + The example structure has been modified to suggest changes that may be made to the .c file. These changes to the structure are required to support changes being made to the public interface
  + The function, BehaviorHudTextSetDisplay(), has been removed
  + The function, BehaviorHudTextRead(), has been added

BehaviorHudText.c

* You must make the following changes to this file for Project 6:
  + BehaviorHudTextCreate
    - Set the score system ID to SsiInvalid
  + BehaviorHudTextRead
    - If both pointers are valid,
      * Call BehaviorRead() to read the base behavior values
      * Read a token and store it as the text formatting string
      * Read an integer and store it as the score system ID
  + BehaviorHudTextUpdate
    - Instead of the original “watchValue”, call ScoreSystemGetValue() and compare the result with “displayValue”
  + BehaviorHudTextUpdateText
    - If the score system ID is not equal to SsiInvalid,
      * Set “displayValue” = ScoreSystemGetValue()
      * Update the display string using the existing code

GameStateAsteroids.h

* You must make the following changes to this file for Project 6:
  + Remove all references to the following function:
    - GameStateAsteroidsIncreaseScore()

GameStateAsteroids.c

* You must make the following changes to this file for Project 6:
  + Remove all references to any existing Asteroid score variables
  + Remove all references to the following functions:
    - GameStateAsteroidsCreateHudElement()
    - GameStateAsteroidsIncreaseScore()
  + GameStateAsteroidsLoad
    - Add a call to ScoreSystemInit()
  + GameStateAsteroidsInit
    - Add code to build the following game objects and add them to the active game object list:
      * AsteroidsScore, AsteroidsHighScore, AsteroidsWave
    - Replace existing score and wave count code with a call to ScoreSystemRestart()
  + GameStateAsteroidsUpdate
    - If the user *triggers* the ‘4’ key, change the game state to the Omega level
  + GameStateAsteroidsSpawnAsteroidWave
    - Replace existing wave count code with a call to ScoreSystemIncreaseWaveCount()

Collider.h

* This header file has been updated as follows:
  + A ColliderType enum has been added
  + The Collider structure has been moved from the .c file to the .h file. This facilitates the creation of new “derived” Collider components
  + ColliderCreate has been removed. It has been replaced by the Create functions in the two new Collider components.
    - [In C++, this would be implemented using a pure virtual constructor and derived constructors.]

Collider.c

The following changes should be made to your existing Collider.c file:

* Remove the Collider structure
* Remove the ColliderCreate function
* In ColliderClone:
  + Modify the call to calloc() to use the value in memorySize, rather than sizeof(Collider) to ensure that the correct amount of memory is allocated
  + Replace the code to copy the contents of “other” with the following:
    - memcpy\_s(clone, other->memorySize, other, other->memorySize);
  + Set the clone’s parent to NULL
* In ColliderCheck:
  + Replace your implementation of the collision detection with a call to a new private function:
    - bool ColliderIsColliding(ColliderPtr collider1, ColliderPtr collider2)
  + If this new function returns true, then invoke the collision handlers, as per the instructions in Project 5
* In ColliderIsColliding:
  + This new, *private* function is responsible for calling the correct collision check function, as determined by the ColliderType of the two colliders
    - If both colliders are circle colliders,
      * Return the result from ColliderCircleIsCollidingWithCircle()
    - If both colliders are line colliders,
      * Return false
      * [Hint: This condition should never be true, as there is only one line collider in the Omega level.]
    - If one collider is a circle and the other collider is a line,
      * Return the result from ColliderLineIsCollidingWithCircle()
      * Make sure to pass the two colliders in the correct order

ColliderCircle.h

* This header file declares the public interface for detecting collisions between two circular objects
* The ColliderCircle structure is “derived” from the Collider struct. Make sure to set the memorySize member variable correctly to avoid severe memory bugs

GameObject.c

* GameObjectRead:
  + For the BehaviorHudText component, replace the call to BehaviorRead() with a call to BehaviorHudTextRead()
  + Add code to handle the ColliderCircle and ColliderLine components. The code to handle the Collider component may be removed

GameStateTable.c/.h

* Add the “Omega” game state to the GameStates enum and the game state lookup table
* Set the initial game state to GsOmega

GameStateLevel1.c

* You must make the following changes to this file for Project 6:
  + GameStateLevel1Update
    - If the user *triggers* the ‘4’ key, change the game state to “GsOmega”

GameStateLevel2.c

* You must make the following changes to this file for Project 6:
  + GameStateLevel2Update
    - If the user *triggers* the ‘4’ key, change the game state to “GsOmega”

GameStateDemo.c

* You must make the following changes to this file for Project 4:
  + GameStateDemoUpdate:
    - If the user *triggers* the ‘4’ key, change the game state to “GsOmega”

GameStateSandbox.c

* You must make the following changes to this file for Project 4:
  + GameStateSandboxUpdate:
    - If the user *triggers* the ‘4’ key, change the game state to “GsOmega”

GameStateOmega.c/.h

* **It is *highly recommended* that you complete the code refactoring outlined above before implementing this step**
* You will need to create these files by duplicating GameStateAsteroids.c/.h.
  + Rename the function names to replace “GameStateAsteroid” with “GameStateOmega”
* You will also need to make the following changes to these files for Project 6:
  + GameStateOmegaInit
    - Replace the “Spaceship” game object with the “SpaceshipOmega game object
    - Create an “Arena” game object and add it to the game object manager’s active list
    - Replace the existing HUD Text objects with:
      * OmegaScore, OmegaHighScore, OmegaWave
  + GameStateOmegaUpdate
    - If the user *triggers* the ‘1’ key, change the game state to Level 1
    - If the user *triggers* the ‘2’ key, change the game state to Level 2
    - If the user *triggers* the ‘3’ key, change the game state to the Asteroids level
    - If the user *triggers* the ‘4’ key, restart the current level

Data/MeshArena.txt

* This data file specifies the mesh data for the rectangular inner arena wall sprite
  + **Requirement:** This file must be created by you
* The mesh may be created using a quad mesh or a list of vertices
  + **Requirement:** The vertices must fit within a unit-sized area positioned around the origin (-0.5f to 0.5f)
  + **Requirement:** The mesh color must contrast with the color in the sprite font sheet. If the HUD text is difficult to read because of poor color choice, then a penalty may be applied

Data/MeshAsteroid.txt

* This data file must be copied over from your Project 5 implementation

ColliderLine.h

* This header file declares the public interface for detecting collisions between a line collider and a circle collider
* The ColliderLine structure is “derived” from the Collider struct. Make sure to set the memorySize member variable correctly to avoid severe memory bugs

ColliderLine.c

* Implement point-line collision detection, as discussed during the Week 9 lectures (Week 7 during the Summer semester). Refer to the lecture slides, available on Moodle, for implementation details
* Implement reflection, as discussed during the Week 9 lectures (Week 7 during the Summer semester). Refer to the lecture slides, available on Moodle, for implementation details. Each object’s velocity and rotation should be updated correctly following the collision/reflection
* NOTE: You are not required to implement circle-line collision (radius > 0) detection for this Project

# Submission Requirements

* The project must build cleanly, with no errors or warnings.
* Once the assignment has been completed, create a submission .zip file by performing the following steps:
  + Select the following files and folders:
    - “AE” folder
    - “Assets” folder
    - “Data” folder
    - “Source” folder
    - Project6.sln
    - Project6.vcxproj
    - Project6.vcxproj.filters
  + Right-click on one of these files and select the option:
    - “Send to” -> “Compressed (zipped) folder”
  + The resultant .zip file **must not** include any of the following Visual Studio generated folders and files:
    - Folders: “Debug”, “Release”, “ipch”
    - Files (\*.db, \*.sdf, \*.opendb)
  + Rename the resultant .zip file using the following naming convention:
    - CS230SU22<section letter>\_<Login ID>\_Project6.zip
      * Example: CS230SU22A\_john.doe\_Project6.zip
* Upload the submission .zip file via the Moodle page for your CS230 section (A or B)
* Once your submission has been uploaded, it is highly recommended that you verify that the submission process was completed successfully, by performing the following steps:
  + Return to the home Moodle page for your section (A or B)
  + Click on the assignment submission link
  + Download the .zip file to your computer
  + Unzip the contents of the .zip file into an empty folder
  + Open the Visual Studio solution file
  + Clean and rebuild the project
  + Verify that the program runs correctly (within Visual Studio is fine)

# Assignment Grading Guidelines

* A -25% penalty will be applied for each week or portion of a week that the project is submitted late. However, no project submissions will be accepted after end-of-day Friday of Finals Week, without prior written approval of the instructor
* A -10% penalty will be applied to any submissions that are performed incorrectly (e.g. incorrect .zip format, submitting extraneous files, etc.)
* A -10% penalty will be applied to any submissions that do not conform to the naming convention specified in the Submission Requirements section