# Due Date

This assignment must be completed and submitted via Moodle before end-of-day on Saturday during Week 14.

# Objectives

The objectives for this project are as follows:

* Implement a component-based game object design architecture in C++
* Update the previous game states to use this new architecture

# Description

For this project, you will update the previously implemented game states (Level 1, Level 2, Asteroids & Omega) by switching the game object structure to a component-based class using C++’s inheritance and polymorphism, where the game object serves as a container of different components.

For this project, you may elect to exclude the Asteroids and Omega levels for reduced credit, as outlined in the following Rubric section. Any levels that you do implement must be accessible from all other levels (using the ‘1’, ‘2’, ‘3’, and ‘4’ keys). The default level should be set to “Level 1”. Any levels that cannot be accessed from Level 1 will be assumed not to exist.

# Rubric

* Level 1 & 2 (80%)
  + Successfully implement the Level 1 (Monkey) and Level 2 (Cursor Following) functionality after converting the following modules from C to C++:
    - GameObject, GameObjectManager
    - Animation, Sprite, Physics, Transform
* Asteroids (10%)
  + Successfully implement the Asteroids Level functionality after converting the following additional modules from C to C++:
    - GameObjectFactory
    - Behavior, BehaviorSpaceship, BehaviorBullet
  + NOTE: You may omit the asteroids and collision functionality from this level.
* Omega (10%)
  + Successfully implement the Omega Level functionality after converting the following additional modules from C to C++:
    - BehaviorAsteroid
    - Collider, ColliderCircle, ColliderLine

# Suggested Steps

1. Migrate your source code (.c files) from Project 6 to Project 7.
2. Rename all .c files to .cpp (preferably from within Visual Studio). Alternatively, you could remove all .c files from the VS project, rename the files, and add the new .cpp files to the VS project.
3. Change the project to be compiled as C++
   * Right click on the project in Visual Studio
   * Click on “Properties”
   * Next to “Configuration”, select “All Configurations”
   * On the left side, expand the “C/C++” option
   * Click on “Advanced”
   * Next to “Compile As”, select “Compile as C++ Code (/TP)”
   * Click “OK”
4. If necessary, update the project settings for the precompiled header
   * Right click on “stdafx.cpp”
   * Click on “Properties”
   * Next to “Configuration”, select “All Configurations”
   * On the left side, expand the “C/C++” option
   * Click on “Precompiled Headers”
   * Next to “Precompiled Header”, select “Create (/Yc)”
   * Click “OK”
5. Click on “Build/Rebuild Solution”
6. Clean up all compiler warnings and errors
7. Verify that all levels in the project still work correctly

# Files

NOTE: For this project, you may freely add/remove/edit the code and header files. However, you should maintain good programming practices and avoid making radical changes to the project that might make it difficult for TA’s to grade your project.

When migrating header files from C to C++ you will need to remove the following code:

#ifdef \_\_cplusplus

extern "C" {

/\* Assume C declarations for C++ \*/

#endif

#ifdef \_\_cplusplus

} /\* End of extern "C" { \*/

#endif

Component.cpp/.h

The project must have an abstract base component class called “Component”. This class should have:

* A public enum (for example, *TypeEnum*) with IDs for each of the component types:
  + Animation, Behavior, Collider, Physics, Sprite, Transform
* A pure virtual function: virtual ComponentPtr Clone(void) const = 0;
  + Every derived class should implement an overload for the Clone function. Incorrectly cloning components can result in memory-related bugs.
  + *Alternatively, you may implement copy constructors instead of a Clone function.*
* A virtual function for updating components:
  + virtual void Update(float dt)
* A virtual function for drawing components:
  + virtual void Draw() const
* A private variable of type, TypeEnum, which is used to specify the type of the component (Transform, Sprite, etc…)
  + This variable must be set only when a component is constructed
    - *Hint: You are* not *allowed to create an accessor function for* setting *the component type*
  + Implement an accessor function for getting the component type
* A private variable of type, GameObjectPtr (or GameObject \*), which is used to specify the “parent” game object of the component.
  + Implement accessor functions for getting and setting the parent pointer
  + This variable must replace the “parent” variables present in the original Behavior and Collider structures
  + Also, remove the “sprite” pointer present in the Animation structure. The animation code should now access the sprite component by means of this “parent” game object.

GameObject.cpp/.h

* Implement a container for attaching components. This container can be an array or a standard-template library class containing variables of type, ComponentPtr.
* Implement a function to add components to the game object. This function should accept a parameter of type, ComponentPtr (or Component \*).
  + Make sure to set the component’s “parent” pointer at this time
* Implement a function to iterate through the attached components and return a pointer to a component that matches the specified type. Refer to the “Components in C++” lecture slides for suggestions on possible implementation.
* Modify the GameObjectUpdate function to iterate through all attached components and call the component’s Update function.
* Modify the GameObjectDraw function to iterate through all attached components and call the component’s Draw function.
* Modify the GameObjectFree function to free all attached components.

# Submission Requirements

* The project must build cleanly, with no errors or warnings.
* The Visual Studio project files that you submit must be named Project7\*.\*. If you are using a previous set of project files, then please rename them by changing the solution and project properties in Visual Studio. Make sure that the resultant .exe filename is also changed to Project7.
* 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
    - Project7.sln
    - Project7.vcxproj
    - Project7.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:
    - CS230S19<section letter>\_<Login ID>\_Project7.zip
      * Example: CS230S19A\_john.doe\_Project7.zip
* Upload the submission .zip file via the Moodle page for your CS230 section (A, B, or C)
* 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
  + 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

# Assignment Grading Guidelines

* A -25% penalty will be applied to any late submissions. No submissions will be accepted after Friday of Week 15 (Finals Week).
* 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.