# CSC481/581 - Module 8: Object-centric Game Object Models

### Overview

This assignment is designed to give you hands-on experience with an **object-centric game object model**. In this paradigm, each entity in the game world is a self-contained object, typically inheriting from a common base class. The object's class defines both its data (e.g., position, color) and its behavior (e.g., how it moves, how it's drawn).

This template provides a basic game loop using C++ and SDL3. It includes a GameObject base class and a Rectangle class that inherits from it. Your task is to extend this model by creating a new Circle class.

## Learning Objectives

- LO 8.2: Design an object-centric model to represent dynamic entities in a game world.
- LO 8.3: Implement an object-centric game object model for real-time interactions.

## Your Task

Your goal is to implement a new Circle class that inherits from the GameObject base class.

- 1. Create Circle.h and Circle.cpp files.
  - In Circle.h, declare the Circle class, making sure it inherits publicly from GameObject.
  - Add a member variable for the circle's radius.
  - Override the update() and render() virtual functions.
- 2. Implement the Circle class in Circle.cpp.
  - The constructor should initialize the circle's position, velocity, color, and radius.
  - The update() function should implement movement logic. For example, make it move horizontally and bounce off the screen edges.
  - o The render() function should draw a filled circle on the screen. Note: SDL does not have a built-in function to draw a filled circle, so you'll need to implement the logic for this yourself! A simple algorithm is to iterate from x = -radius to x = radius and y = -radius to y = radius, and if x\*x + y\*y <= radius\*radius, draw a point at (centerX + x, centerY + y).</p>
- 3. Integrate the Circle into main.cpp.
  - Include Circle.h.
  - In the main function, create an instance of your new Circle object.
  - Add your Circle object to the gameObjects vector.
  - The existing game loop will automatically call update() and render() on your object.

## **Prerequisites**

PROFESSEUR: M.DA ROS

You must have **SDL3** installed on your system.

- macOS: brew install sdl3
- Linux (Debian/Ubuntu): sudo apt update && sudo apt install libsdl3-dev
- Windows (MSYS2): pacman -S mingw-w64-ucrt-x86\_64-SDL3

## **Project Structure**



## How to Build and Run

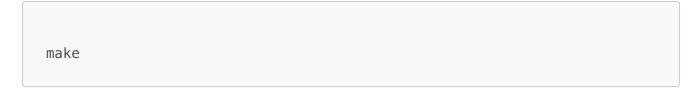
1. Create a Build Directory

```
mkdir build
cd build
```

2. Configure with CMake

```
cmake ..
```

3. Compile the Project



The executable will be created inside the build/ directory.

4. Run the Game

./ObjectCentricDemo

#### Controls

Arrow Keys: For movements

Escape Key or Window Close Button: Quit the application.

### CMakeLists.txt

This build script will find SDL3 and compile the C++ source files into an executable.

```
# CMakeLists.txt
cmake_minimum_required(VERSION 3.10)
project(ObjectCentricDemo)
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
# Use pkg-config to find SDL3. This is the most reliable method on
macOS.
find_package(PkgConfig REQUIRED)
pkg_check_modules(SDL3 REQUIRED IMPORTED_TARGET sdl3)
# Create the executable from your source files
add_executable(
    ObjectCentricDemo
    src/main.cpp
    src/GameObject.cpp
    src/Rectangle.cpp
    # TODO: Add your Circle.cpp file here when you create it!
    # src/Circle.cpp
)
# Link the SDL3 library using the target found by pkg-config
# This automatically handles include directories and linker flags.
target link libraries(ObjectCentricDemo PRIVATE PkgConfig::SDL3)
# Also add our own src directory to the include path for cleaner
#includes
target_include_directories(ObjectCentricDemo PRIVATE
${CMAKE_CURRENT_SOURCE_DIR}/src)
```