# CSS 452: Final Project

## Improving the Game Engine

**Due time:** Please refer to our course web-site

### Objective

As the last project of our learning, we will examine the engine we are learning from the perspective of supporting an existing 2D videogames, identify the shortcomings and/or inadequacies of our game engine, propose appropriate functionalities to improve the game engine, define the API for the proposed functions, perform a prototype implementation, and, demonstrate your new functionality in simple game-like examples.

This is a group assignment. Each group should be 2 or 3 persons. There can be no single person team.

The following are the requirements:

### Goals of the Final Project:

The goals are to demonstrate your ability to:

- Identify functionality that belongs to the game engine vs specific-games
- Describe and design an appropriate API supporting the functionality
- Implement a prototype verifying the functionality
- Work with moderate code and functionality complexity

With the objectives to accomplish the above goals, here is a list of technical specification.

### Final Project Technical Specifications:

Your final project deliverables include both elements during the development process and a final Software Development Kit (SDK).

1. **Survey and Identify Missing Functionality**:
    a. You will
        i. Survey and find existing 2D game(s)
        ii. Describe the underlying engine support requirements
        iii. Compare to our own game engine (with functionality at the end of Example 7.5)
        iv. Identify required functionality that is lacking from our game engine
    b. Deliverable
        i. A 5 to 7 minute presentation, overviewing the 2D game(s), highlighting the missing functionality, and describing the desirable integration into our game engine

2. **Propose an API to deliver the Missing Functionality**:
    a. You will
        i. Formally define the missing functionality (e.g., Camera Behavior, Hero movement in platformer)
        ii. Define the required Application Programming Interface (API), or set of classes and behavior, that is required to support the functionality
        iii. Show (without implementation) sample API calls in a game that will accomplish the functionality
    b. Deliverable:
        i. A 5 to 7 minute presentation, defining the functional module you will develop and deliver, list of the API functions you must implementing, and including samples of how the API classes/functions will be used
        ii. A 2-4 page report including the draft API documentation and code fragments showing how the functions will be invoked.

iii. **Must include:** Example user code showing how to call the API from (which ever applicable):
- MyGame::MyGame (constructor, *if any*)
- MyGame::initialize()
- MyGame::loadScane() [ *if any* ]
- MyGame::update()
- MyGame::draw() and
- MyGame::unloadScene() [ *if any* ]

c. Hint: An API that is reasonable for the final project time frame (2-3 weeks) would probably include between 5 to probably less than 10 functions. We can use SpriteAnimate functionality/module as an example.
   i. **Initialization of state**: classes and/or functions that are meant to be called from within the MyGame::initialize() function to initialize/configure internal state. E.g., The SpriteAnimate functionality expects the user to identify the sprite elements and configure initial animation speed in the MyGame::Initialize() function.

   ii. **Update of state**: classes/functions that are meant to be called from within the MyGame::update() function to cause updates of your internal state. There are two main categories:
      1. **Internal state update**: your API would probably need to update its internal state constantly. This would probably be a single function call, e.g., SpriteAnimate system defines *updateAnimation()* function to update module internal state.

      2. **Player interaction support**: if your module expects input/interactions form the user, you will have functions that can be triggered directly by the users to cause state update. E.g., SpriteAnimate system expects user to change animation type and increase/decrease animation rates, and thus, the module supports *SetAnimationType()*, and *incAnimationSpeed()* user interaction support.

3. **Progress demo of API functionality**:
   a. You will
      i. Demo of your progress in implementing your API
      ii. Show and explain how API are invoked in the source code of your demo
      iii. Present at least two use cases (to enhance confidence that there is an API)
   b. Deliverable:
      i. 5-7 minutes presentations

4. **Final demo of your API implementation and deliver a mini-SDK**
   a. You will
      i. Demo at least two different games each invoking your API
      ii. Explain your source code
      iii. Show your API documentation
   b. Deliverable:
      i. 5-7 minutes presentation of presentation
      ii. A 4-6 page SDK documentation including:
         1. Module functionality description: 1 to 2 paragraphs clearly describe your module
         2. API documentation: e.g., here is the API documentation for our game engine (https://gamethemedgroup.github.io/GTCS-GameEngine/AdditionalMaterials/EngineAPIDocumentation/docs/index.html)
         3. Tutorial (explain how to use your API): e.g., here are some tutorials (https://gamethemedgroup.github.io/GTCS-GameEngine/AdditionalMaterials/EngineAPITutorials/index.html)for our game engine API
         4. **Conclusion:** evaluate your implementation: strengths and weaknesses
            a. If you had more time, what would you have done? What should the next version do?

The final 20% of your project will be evaluated accordingly:

| **What** | **From Kelvin (18%)** | **From Peers (2%)** |
|---|---|---|
| Form a 2-3 person team | 0.5% | |
| Survey of Missing Functionality | 1% | 0.5% |
| Proposal Presentation<br>Proposal report<br>Draft listing of API classes/functions<br>Sample code fragment of using API<br>**Rubric:**<br>➔ Description of key classes: +2<br>➔API functions with proper comments: +2<br>➔Example of user code: Make sure to specify the calls from<br>MyGame::Initialize(): +2<br>MyGame::Draw(): +2<br>MyGame::Update(): +2 | 1%<br><br>1.5% | 0.5% |
| Progress Presentation | 2% | 0.5% |
| Final Presentation<br>SDK Documentation + Source code submission<br>Proper functioning demo samples (at least two), including<br>**Context:** your names, UWB, CSS452<br>**Instructions**: how to run the examples<br>SDK must be in HTML format and appear professional<br>Module description<br>Your changes (changed files in the engine folder system)<br>API documentation<br>Tutorial<br>API Complexity and Completeness<br>Implements the functions proposed<br>API Quality:<br>Reasonable abstraction with classes<br>Reasonable function and variable names<br>Minimal global state (variables and functions) | 2%<br><br>4%<br><br>2%<br><br>2%<br><br>2% | 0.5% |

The final project counts 20% towards your final grade for this class.