

Kielce University of Technology
Faculty of Electrical Engineering, Automatic Control and
Computer Science

Computer graphics

Basics of the computer graphics

Laboratory instruction 1

”**Engine** class
Main game loop
Keyboard and mouse support”

Authors:
Paweł Pięta, MSc
Daniel Kaczmarek, MSc

Translated by:
Konrad Sich, MSc

Kielce, 2023

1 Introduction

The aim of the classes is to introduce the fundamental class of the game engine and outline its responsibilities, and then implement such functionalities as initializing the graphics library, the main game loop, keyboard and mouse support, and error handling.

2 Engine class

Typically, the heart of any game engine, the foundation of a video game, is a class called **Engine**. This class typically implements the following functionality:

- initialization of the graphics library,
- management of game objects,
- main game loop support,
- support for input devices (keyboard, mouse, etc.),
- error handling,
- clearing memory before closing the game.

Of course, the above list does not exhaust the functions that the **Engine** class may be responsible for. It is also worth mentioning that this class is often implemented using the **singleton** design pattern – it is reasonable that only one object of this class exists in memory while a video game is running.

2.1 Initialization of the graphics library

To start working with the graphics library, it must be initialized appropriately. Appropriate functions described in the documentation of a given library are used for this purpose. You should also remember to check the correctness of the initialization (e.g. based on the value returned by the initialization function). If initialization is successful, you can proceed to launch the game menu or game loop. Otherwise, you can consider re-initializing, e.g. for modified configuration parameters, or logging an appropriate error message to the file and terminating the game engine.

2.2 The main game loop

The most important function of the game engine is to support the main game loop. The following operations should be performed in this loop:

- support for input devices,
- updating the game's logical layer (including artificial intelligence calculations),
- updating the physical layer of the game (including collision checking),

- updating the game’s graphic layer (rendering),
- update of the game’s sound layer.

In the context of a game loop, a frequently used thing is the timer. This is an event source that generates events at regular intervals – they are used to regulate the running time of the code. In video games, the primary use of timers is to trigger the main game loop. There is nothing stopping the game from using several timers that will be responsible for triggering updates to various aspects of the game’s logical layer.

Since the first step in updating the game’s graphics layer is usually clearing the screen bitmap (e.g. to a given color), it is worth equipping the engine with a method that facilitates such an operation. Ultimately, the **Engine** class should make it easier to clean not only the screen bitmap, but also any bitmap.

2.3 Keyboard and mouse support

The basic aspect that distinguishes a video game from a movie is its interactivity. However, it cannot exist without implementing support for input devices such as a keyboard, mouse, or even a pad in the game engine. Therefore, the **Engine** class should be equipped with appropriate methods to facilitate reading the status of input devices. It is also worth mentioning that the observer design pattern is often used in this context.



Figure 1: Example input devices: keyboard and mouse

3 Tasks

1. Create a 2D engine framework project.
2. Implement the **Engine** class with the following functionality:
 - initialization of the graphics library,
 - parameterizing the graphic mode (e.g. windowed, full screen, resolution selection, etc.),

- parameterizing other parameters (e.g. number of animation frames per second, starting mouse/keyboard support, etc.),
- main game loop using a timer,
- keyboard and mouse support,
- support for cleaning the screen (and ultimately any bitmap) to a given color,
- error handling with the ability to display or log to a file of diagnostic messages,
- closing the game (deinitializing the graphics library, cleaning up memory, etc.).