Exceptions in C++

Having standards for handling errors is important specially as programs grow and libraries get used more.

I'm making use of the capacity for exceptions to unwind the function call stack to have a single catch function for many throws.

I also learned that is important to use `std::exception` derived types. For WinAPI there is `std::system_error`

`std::exception` → `std::runtime_error` → `std::system_error`

`main.cpp`

```
    try
    {
        App app{ hInstance, nCmdShow };
        statusReturn = app.run();
    }
    catch (const std::system_error& e)
    {
        std::string error_str = std::format("Caught system_error with code {} mea
ning {}",
                            e.code().value(),
                            e.what());
        //std::string error_str = std::format("Caught system_error with code {}" ,
e.what());

        MessageBox(nullptr, error_str.c_str(), "Error!", MB_ICONEXCLAMATION |
MB_OK);
        statusReturn = 1;
    }
    catch (const std::exception& e)
```

```
    {
        MessageBox(nullptr, e.what(), "Error!", MB_ICONEXCLAMATION | MB_O
K);
        statusReturn = 1;
    }
```

app.cpp

```
    // Registering the window
    if (!RegisterClassEx(&wcex)) {
        DWORD ec = GetLastError();
        throw std::runtime_error("RegisterClassEx failed (code " + std::to_string(e
c) + ")");
    }

    ...

        if (!windowHandle_){
        DWORD ec = GetLastError();
        throw std::system_error(static_cast<int>(ec), std::system_category());
    }
```

For other exceptions I can also use

```
std::runtime_error("CreateWindow failed(code" + std::to_string(ec) + ")");
```

RAII

Resource Acquisition Is Initialization, the idea here is for the constructor to acquire all the resources for a class and the destructor release all of those resources. This way the release of class resources is guaranteed

I’m still pending to learn the `Rule of 3` and the `Rule of 5`

https://en.cppreference.com/w/cpp/language/rule_of_three.html

Resources

- A tour of C++ , Pages 43-51

  A tour of C++