Readme: This project is licensed under the terms of the MIT license.

Author: Parcode

**0.Summary**

1.Creating a Window

1. ~ symbol is the destructor and it gets called to free up memory
2. **WNDCLASSEX** structures: [MicrosoftWindowsLink](https://learn.microsoft.com/en-us/windows/win32/api/winuser/ns-winuser-wndclassexa)
   1. **cbCLSExtra** - allocates extra memory for the class structure itself meaning all windows get affected
   2. **cbSize** - setting the size of data for corruption check [StackOverflowLink](https://stackoverflow.com/questions/189622/what-is-the-purpose-of-the-cbsize-member-in-win32api-structs)
   3. **cbWndExtra** - allocates extra memory for each instance of a window meaning individual windows are allocated
   4. **hbrBackground** - defines the background color or pattern for the window
      1. **(HBRUSH)COLOR\_WINDOW** - sets the background of the window to the default system window color
   5. **hCursor** - determines the mouse pointer's appearance
      1. **LoadCursor(NULL, IDC\_ARROW)** - loads the standard arrow cursor
   6. **hIcon** - small image at the top left side of the title bar
   7. **hIconSm** - appearance when pressed Alt-Tab
   8. **hInstance** - differentiates itself from other applications
   9. **lpszClassName** - identifies and creates windows based on this class
      1. **L** - creates a wide-character string (Unicode string)
   10. **lpszMenuName** - tells which menu to use for that class
   11. **style** - define the initial style of windows created from the window class
   12. [RegisterClassEx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms633587.aspx) - registers a  [*window class*](https://msdn.microsoft.com/en-us/library/windows/desktop/ms632596.aspx): [StackOverflowLink](https://stackoverflow.com/questions/31930556/what-is-the-difference-between-wndclassex-and-createwindow)
       1. & - passes a pointer by taking the memory address of **wc**
   13. **CreateWindowEx** - creates a new window by using:
       1. **dwExStyle** - extended window style
       2. **lpClassName** - window class name
       3. **lpWindowName** - window title
       4. **dwStyle** - window style
       5. **x, y** - position
       6. **nWidth**, **nHeight** - size
       7. **hWndParent**- parent window
       8. **hMenu** - menu handle
       9. **hInstance** - application instance
       10. **lpParam** - user-defined data
   14. **WS\_EX\_OVERLAPPEDWINDOW** - includes features such as a title bar, a system menu, minimize and maximize buttons, and a close button
3. add **m\_hwnd** to **CreateWindowEX()**
4. **ShowWindow** - shows the window
   1. **SW\_SHOW** - shows the window in its normal state
5. **UpdateWindow** - forces window to repaint itself immediately
6. **DestroyWindow** - closes the window
7. **LRESULT** - returns various types of information such as messages of errors
8. **CALLBACK** - specifies how the function is called and how its parameters are passed
9. **lpfnWndProc** - handles messages sent to a window of this class
   1. **UINT** - unsigned integer meant to store positive values only
   2. **WPARAM** - word parameter meant for passing messages
   3. **LPARAM** - long parameter meant for passing
10. **switch** - control structure used to make decisions
11. **WM\_CREATE** - message is sent by the system to notify the window about its creation
12. **break** - exits out of the current control structure
13. **WM\_DESTROY** - sends message when the window is about to be closed
14. **PostQuitMessage** - posts a message that signals the application to exit
    1. **0** - indicates a normal application termination
15. **default** - used when no other option is selected
16. **DefWindowProc** - default behavior for window messages
17. **Window** - refers to a rectangular on-screen area
    1. **\*** - used to declare pointers
18. **nullptr** - represents a null or empty pointer
19. **this** - refer to the current object or instance within a class
20. **public** - accessible from any part of the program
21. **virtual** - base class which lets objects manipulate themselves by using that base class
22. **void** - code that doesn’t have a type
23. **->** - used to access members of an object or a structure through a pointer
24. **while** - continuously runs a code as long as the condition is true
25. **PeekMessage** - checks messages without removing them
    1. **PM\_REMOVE** - message should be removed
26. **TranslateMessage** - processes keyboard or mouse input from the user
27. **DispatchMessage** - sends the processed inputs to the window
28. **Sleep** - how long the program should pause
    1. **0** - milliseconds

2.Creating the GraphicsEngine

1. **ID3D11Device** - creates and manages graphics resources related to 3D rendering
2. **D3D\_FEATURE\_LEVEL** - determine the available capabilities of the GPU
3. **ID3D11DeviceContext** - specifies how 3D objects are drawn to screen
4. **D3D\_DRIVER\_TYPE** - specifies the type of graphics driver to be used
5. **D3D\_DRIVER\_TYPE\_HARDWARE** - uses the hardware-accelerated graphics driver
6. **D3D\_DRIVER\_TYPE\_WARP** - renders on CPU when there isn't a dedicated GPU
7. **D3D\_DRIVER\_TYPE\_REFERENCE** - tests compatibility with other hardware
8. **ARRAYSIZE** - calculates the number of elements in an array
9. **HRESULT** - checks whether the function succeeded or failed
10. **D3D11CreateDevice** - creates a Direct3D device to render 3D graphics
11. **SUCCEEDED** - checks if function succeeded
12. **FAILED** - checks if function failed

3. Creating the SwapChain

1. **IDXGIDevice** - for rendering graphics and managing rendering resources
2. **IDXGIAdapter** - queries information about the the GPU's capabilities
3. **IDXGIFactory** - allows DXGI objects to be created and managed
4. **friend** - declared function or class can access private or protected members
5. **\_\_uuidof** - returns a universally unique identifier
6. **\*\*** - accesses the origin of the value
7. **&** - gets the memory address
8. **IDXGISwapChain** - stores the rendered data before presenting it to an output
9. **ZeroMemory** - clear the memory by setting it to 0
10. **DXGI\_SWAP\_CHAIN\_DESC** -
    1. **BufferCount** - Specifies the number of back buffers in the swap chain
    2. **BufferDesc** - back buffer's properties, such as its format, width, and height
       1. **Width** - width in pixels of the back buffer
       2. **Height** - height in pixels of the back buffer
       3. **Format** - defines how color information is stored for each pixel
       4. **RefreshRate** -
          1. **Numerator** - the number of refreshes per second
          2. **Denominator** - the amount of times numerator reaches max
    3. **BufferUsage** - purposes back buffer for rendering or presenting
    4. **OutputWindow** - window where graphics will be presented
    5. **SampleDesc** - specifies the multi-sampling settings
       1. **Count** - how many points are used for each pixel
       2. **Quality** - the quality level for multi-sampling or anti-aliasing
    6. **Windowed** - boolean for fullscreen or not
11. **CreateSwapChain** - creates a swap chain to manage graphics on window
12. **DXGI\_FORMAT\_R8G8B8A8\_UNORM** - common pixel format RGBA
13. **DXGI\_USAGE\_RENDER\_TARGET\_OUTPUT** - uses resources as a render target
14. **delete** - used to deallocate that memory and calls the destructor for objects
15. **LPCREATESTRUCT** - accesses information about creating or modifying windows
16. **GWL\_USERDATA** - accesses or sets user-specific data associated with a window
17. **GetWindowLong** - retrieves long integer values associated with a window

4. Creating the DeviceContext

1. **ID3D11RenderTargetView** - finds the required resources for rendering
2. **ID3D11Texture2D** - is an object that stores a flat image