3D Graphics API

Libraries we use to draw

Overview

Low-Level API

- Direct3D
- OpenGL
- Vulkan
- Metal

High-Level API

- OGRE
- Java 3D
- Irrlicht Engine
- Three.js

Direct3D

subset of DirectX

Developer: Microsoft

Version: 12

Windows only

OpenGL

Developer: Kronos Group

Version: 4.5

Platform independent

Advantages

DirectX

- Object Oriented
- single SDK
- well documented
- many tutorials

OpenGL

- platform independent
- simple to learn
- embedded systems

Disadvantages

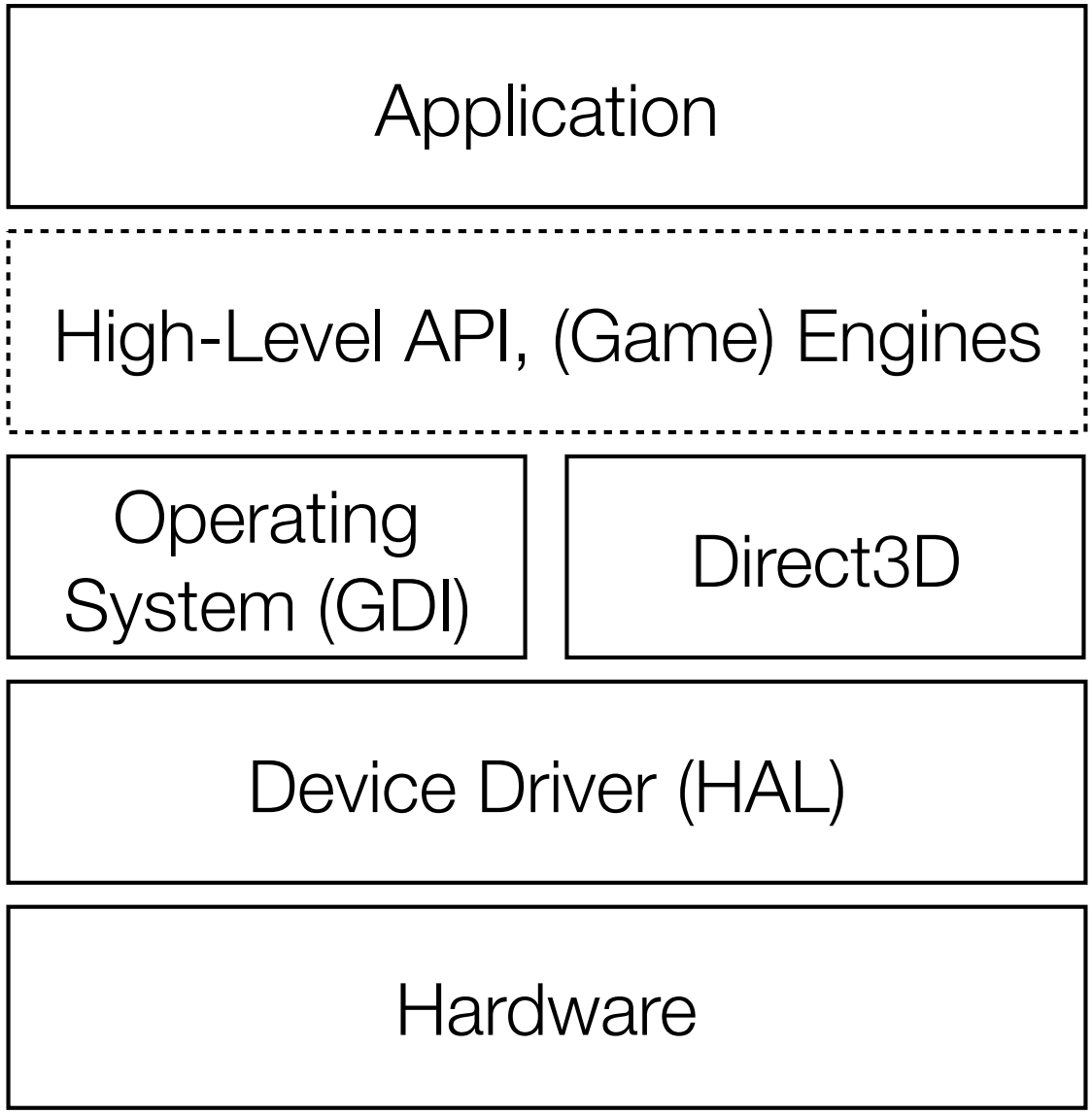
DirectX

- Windows only
- difficult to learn
- not all versions for all Windows versions

OpenGL

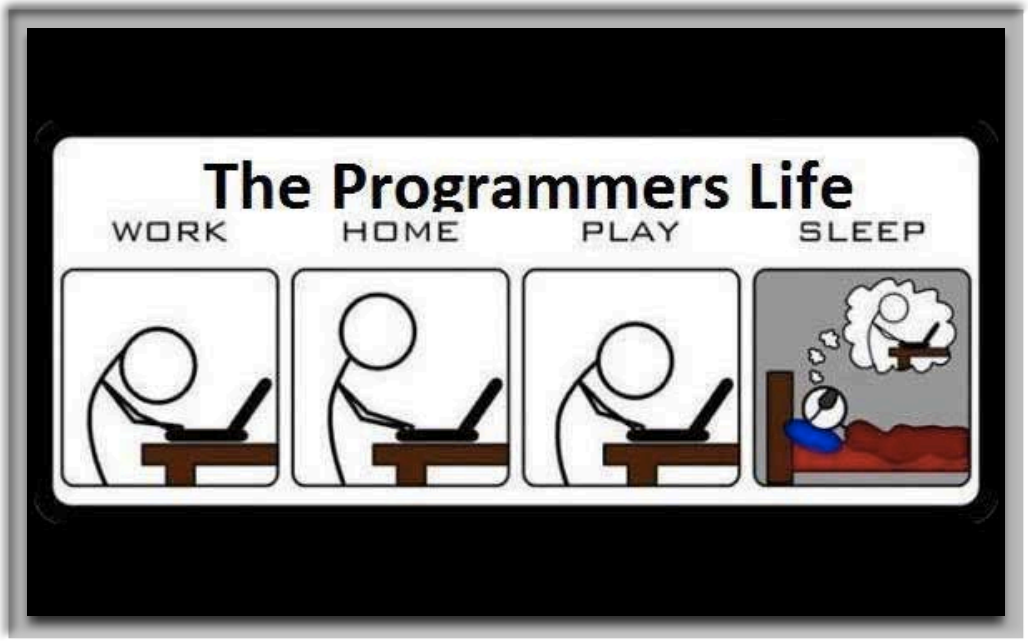
- lot of extensions
- old literature and tutorials
- no good software design because of simple functions

Abstract Layer



Coding Time

Let’s create a window



Window

every application needs a window

graphic accelerated content needs also a window

window will be created with the help of Windows-API

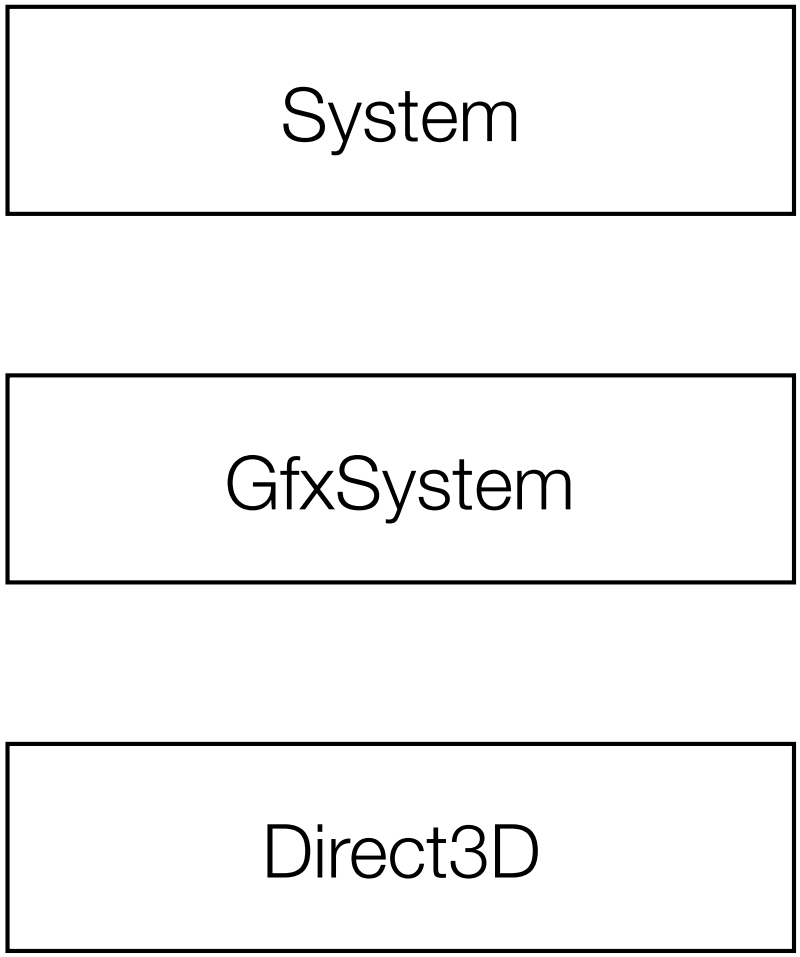
communication between application and Windows with the help of messages

Direct3D

many objects are necessary

- ID3D11Device
- ID3D11DeviceContext
- IDXGISwapChain
- ID3D11RenderTargetView
- ID3D11RasterizerState
- ID3D11DepthStencilView and ID3D11DepthStencilState

Program structure



Direct3D I

ID3D11Device

connection to graphic card capabilities

need for creating other classes for rendering pipeline

ID3D11DeviceContext

connection to graphic card rendering pipeline

need for rendering all objects

Direct3D II

IDXGISwapChain

connection to backbuffers

need for swichting between front- and backbuffer

ID3D11RenderTargetView

connection to a texture (mostly backbuffer)

target during rasterization in rendering pipeline

Direct3D III

ID3D11DepthStencilView

connection to a texture like ID3D11RenderTargetView

texture not for presenting, saves data

Depth

Stencil Mask

ID3D11DepthStencilState

information how and when depth and stencil mask will be saved

can be switched during app cycle

Direct3D IV

ID3D11RasterizerState

information about rasterization process

can be changed during app cycle

Viewport

space for rendering frontbuffer in app frame

mostly one, more than one for split screen

Front- and Backbuffer

both are large memory dumps for saving pixel data

size depends on resolution and pixel format

frontbuffer

- actual presented picture

backbuffer

- actual buffer graphic card is writing on
- at least one

Random

rand()

- generates integral numbers
- use srand() for setting a seed
- old variant

<random>

- huge library for random numbers in c++
- generators
- distributors

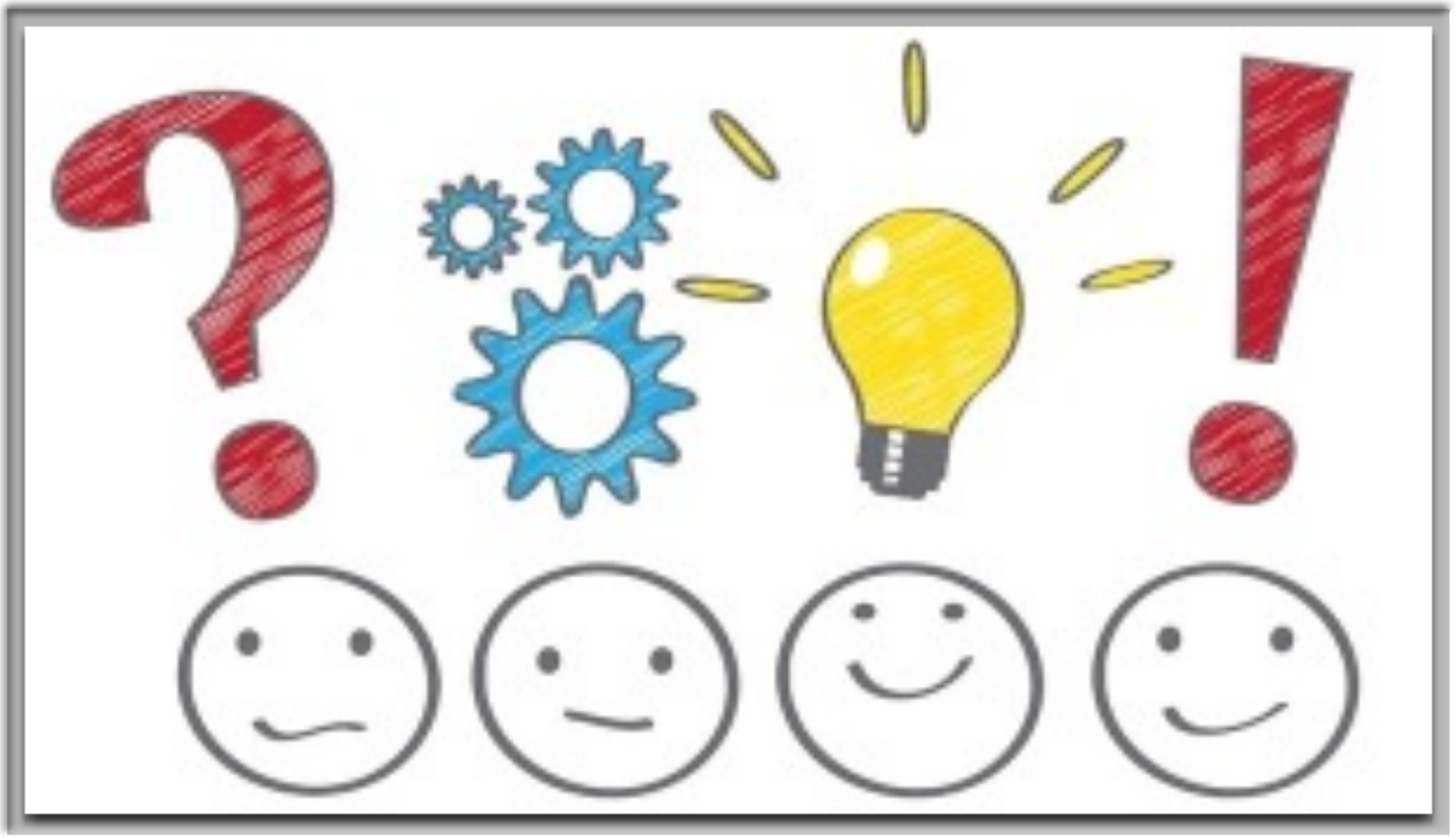
High Frequency Timer

realtime applications needs a precise time

time calculation based on cpu tick

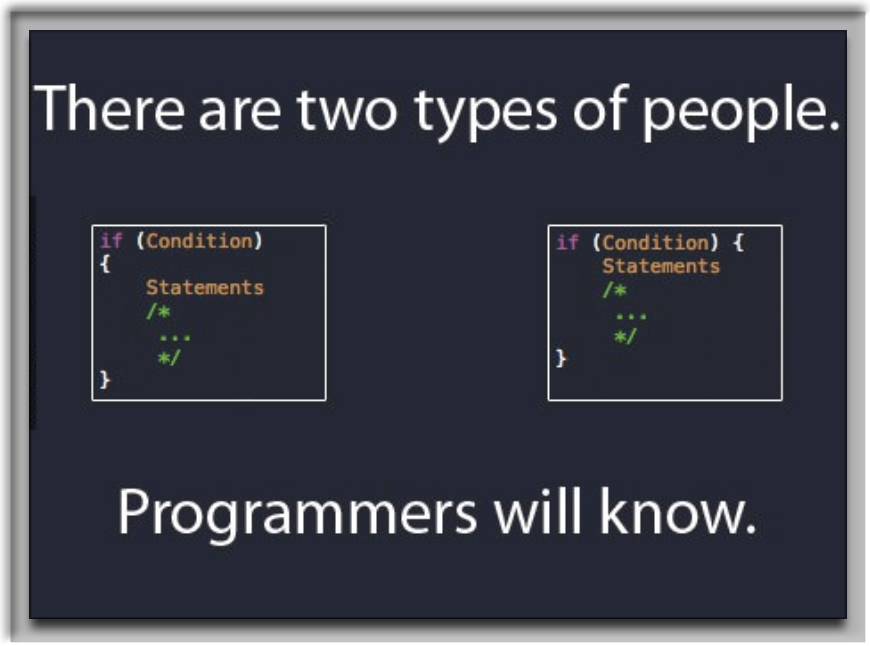
GetPerformanceFrequency()

GetPerformanceCounter()



Coding Time

Let’s create a
connection to Direct3D



https://github.com/TheHacky/GDP2019.git