pipeline object
holds shaders and configuration
VkPipeline: Holds the state of the GPU needed to draw. For example: shaders, rasterization options, depth settings
A set of **VkPipeline** objects can be used for shader combinations and parameters needed to render materials

VkCommandPool objects are used to allocate command buffers

render pass
VkRenderPass: Holds information about the images you are rendering into. All drawing commands have to be done inside a render pass. (main pass, shadow pass)

Math Affine Transformation
Linear Transformations Scaling, Rotation around origin or anchor in 3x3 matrix, plus 4x1 translation (x,y,z,h coordinates)
h=1 for positions, h = 0 directions, distances, which are affected by scale&rotation, not translation
Euler angles will be used: Y(1), X(2), Z(3)
Quaternions have a few advantages over matrices (expressing 3D rotations)
Canonical view volume: normalized 2x2x1 **Vulkan:** y axis points down, z into screen (right hand) **Orthographic projection matrix:** scaled and translated from canonical volume, z useless except for depth test
Perspective projection: square frustum, orthographic matrix * perspective matrix
Viewport: specifies target framebuffer
Model matrix: TRS (from right to left: scale, rotate, translate) **Model-View-Projection matrix:** from left to right

command buffer
work GPU has to be recorded into a CommandBuffer, then submitted into a Queue
One command buffer / queue for each swap chain image
VkCommandBuffer: GPU commands
VkQueue: queue for commands

VkInstance: The Vulkan context, used to access drivers

VkPhysicalDevice: A GPU. Used to query physical GPU details, like features, capabilities, memory size

VkDevice: The “logical” GPU context to execute commands on

VkFrameBuffer: Holds the target images for a render pass

A descriptor is a pointer to a resource (buffer or image). **VkDescriptorSet:** Holds the binding information that connects **shader inputs** to data such as **VkImage** textures and **VkBuffer** resources; a set of gpu-side pointers to be bound once. Descriptors must be grouped into sets, only sets can be bound.

swap chain (list of images) display frame to the screen
VkSwapchainKHR: Holds the images for the screen. It allows you to render things into a visible window.
Has **VkQueue** handles

VkImage: A texture you can write to and read from (texture data)
VkImageView: representing contiguous ranges of the image sub-resources and containing additional metadata

VkBuffer: GPU visible memory (vertex data)

VkDescriptorSetLayouts:
Blueprint for descriptor set, provided at pipeline creation
vKDescriptorPool: pre-allocated memory for descriptors, provided at pipeline creation

VkSemaphore:
Synchronizes GPU to GPU execution of commands. Used for syncing multiple command buffer submissions one after another.

VkFence: Synchronizes GPU to CPU execution of commands. Used to know if a command buffer has finished being executed on the GPU

Render Loop
- request from the VkSwapchainKHR an image to render to (imageIndex)
- allocate a VkCommandBuffer from
  - a VkCommandBufferPool
  - reuse an already allocated command buffer that has finished execution
- write commands into command buffer
  - start a VkRenderPass (render into the image from swapchain, framebuffer[imageIndex]
    for each object {
    + bind a VkPipeline
    + bind VkDescriptorSet resources for the shader parameters)
    + bind the vertex buffers
    + execute a draw call }
  - end the VkRenderPass
- end the VkCommandBuffer
- submit the command buffer into the queue for rendering
- present the image to the screen
- use a semaphore to make the presentation of the image wait until rendering is finished