VULKAN

Tin Vujasinović

SADRŽAJ

- Uvod
- Prednosti
- Nedostatci
- Razvojno okruženje
- Shaderi
- Primjeri

UVOD

- Što je Vulkan?
  - Moderni 3D grafički API
  - Vulkan je nasljednik OpenGL standarda
  - Razvijen je od strane Khronos group
- Objavljen je 2016. godine
- Dostupan je za razvoj na Windows, Mac i Linux operacijskim sustavima
- Potreba za novijim API javlja se jer stariji API nisu mogli pratiti razvoj hardvera



UVOD

- Koji problemi se javljaju kod korištenja OpenGL-a?
  - Programski model nije ujednačen sa modernim hardware-om (GPU)
  - Jako je zahtjevan za CPU – praćenje ovisnosti, validacije stanja
  - Javljaju se različiti bugovi i neželjene pojave na različitim GPU
  - Jedno jezgreni – ne može iskoristiti puni potencijal više jezgrenih procesora

PREDNOSTI (1/3)

- Cross-platform
  - Ne sadrži pozive prema biblioteka za prikaz
  - Za prikaz na ekranu računala potrebno je uključiti third-party biblioteku
- Standardni *bytecode* oblik shadera
  - GLSL se prevodi pomoću kompajlera u SPIR-V *bytecode*
- Manja umiješanost upravljačkih programa grafičkih kartica

PREDNOSTI (2/3)

- Princip eksplicitne kontrole
  - Daje se ‘obećanje’ da se programer upravljačkom programu dati sve potrebne informacije o tome što želi napraviti
  - Pruža sve potrebne detalje o programu
  - Definira se vrijeme kada će se upravljačkom programu prenijeti naredbe

  - Zauzvrat upravljački program ‘obećaje:
    - Da će izvršiti što programer od njega traži
    - Kada on to traži
    - I to će izvršiti brže nego stariji API

PREDNOSTI (3/3)

- Validation layers (*slojevi validacije*)
  - Vulkan API kreiran za visoke performanse i minimalno korištenje upravljačkih programa
  - Općenito to znači da je debuggiranje i provjeravanje grešaka ograničeno
  - Zbog toga se koriste slojevi validacije koji se uključuju između upravljačkih programa i Vulkan API
  - Pružaju dodatne provjere (upravljanje memorijom, provjeravanje parametara funkcija, …)

NEDOSTATCI

- Strma krivulja učenja
- Zahtjeva prethodno znanje o računalnoj grafici, nije za početnike
- Zahtjeva modernu grafičku procesnu jedinicu koja podržava Vulkan API
  - NVIDIA (https://developer.nvidia.com/vulkan-driver),
  - AMD (https://www.amd.com/en/technologies/vulkan),
  - Intel (https://software.intel.com/)
- Iskustvo razvoja u C++ jeziku, kompajler sa C++17 podrškom
- Za iscrtavanje jednostavnog trokuta potrebno je oko 1000 linija koda

RAZVOJNO OKRUŽENJE

- Vulkan podržava razvoj na Windows, Linux i Mac operacijskim sustavima
- Primjer: Windows
- IDE: Visual Studio 2019
  - Notepad++
- Third-party biblioteke: GLFW (prikaz na ekranu), GLM (linearna algebra)

POSTAVKE

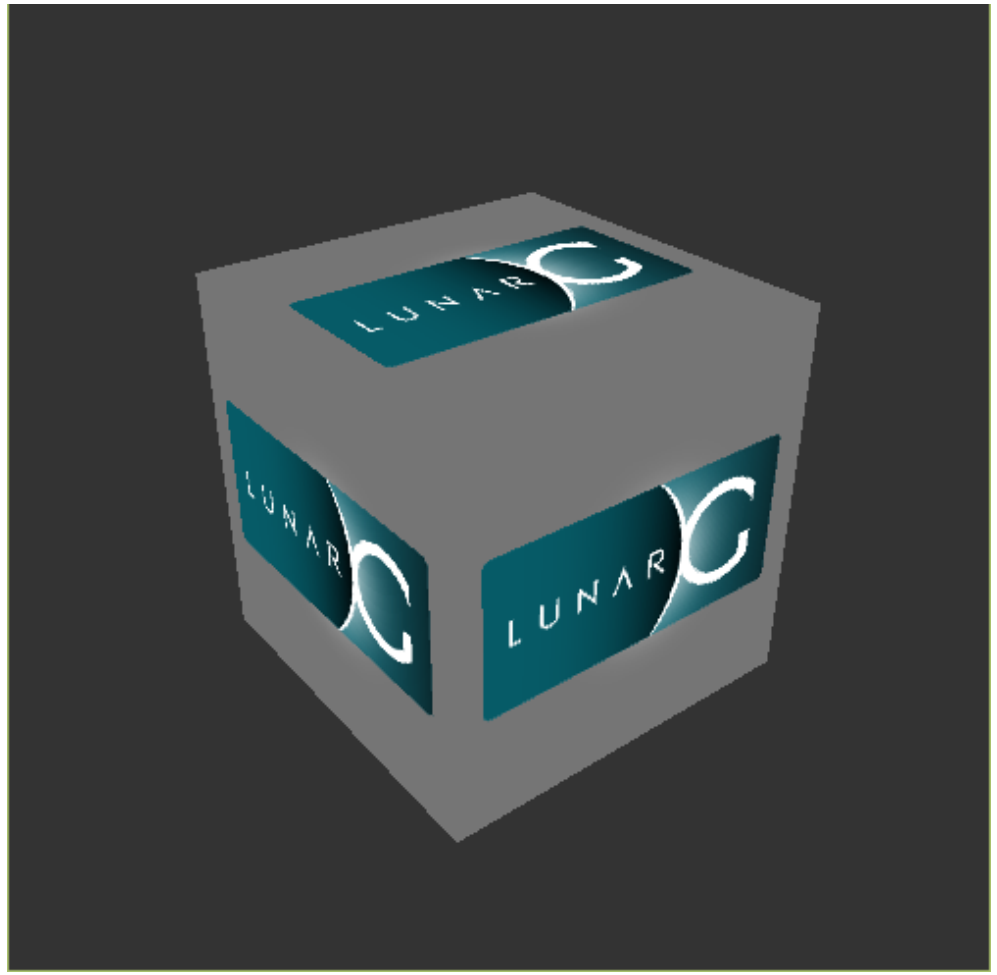
- Potrebno je preuzeti:
  - Vulkan SDK
  - GLFW biblioteku za Windows
  - GLM biblioteku

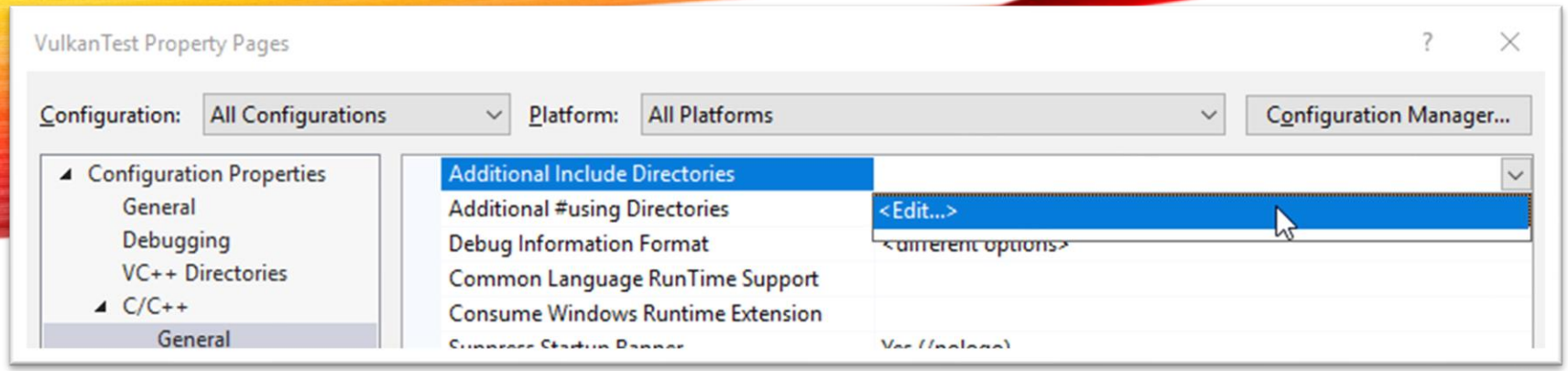
- Postavke Visual Studio 2019
  - C/C++ -> General – Additional Include Directories (header datoteke)
    - (SDK, GLM, GLFW)
  - Linker -> General – Additional Library Directories (object files datoteke)
    -> Input – Additional Dependencies (vulkan-1.lib, glfw3.lib)
  - C/C++ -> Language – C++ 17 standard

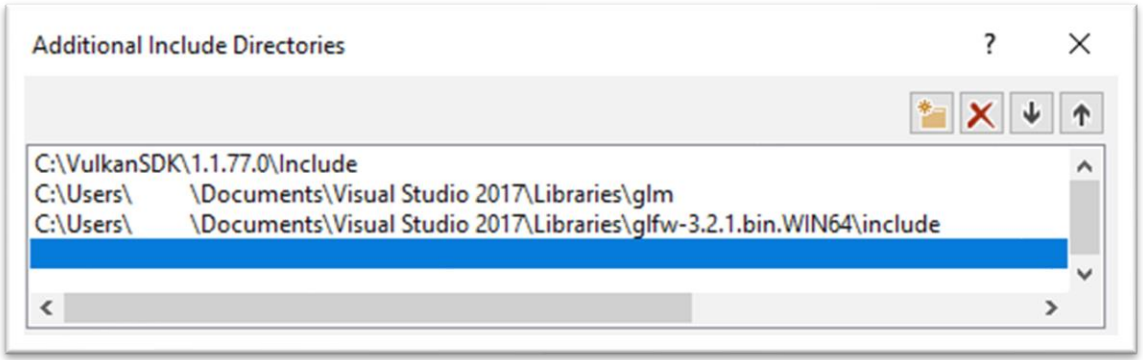
TESTNI PROGRAM

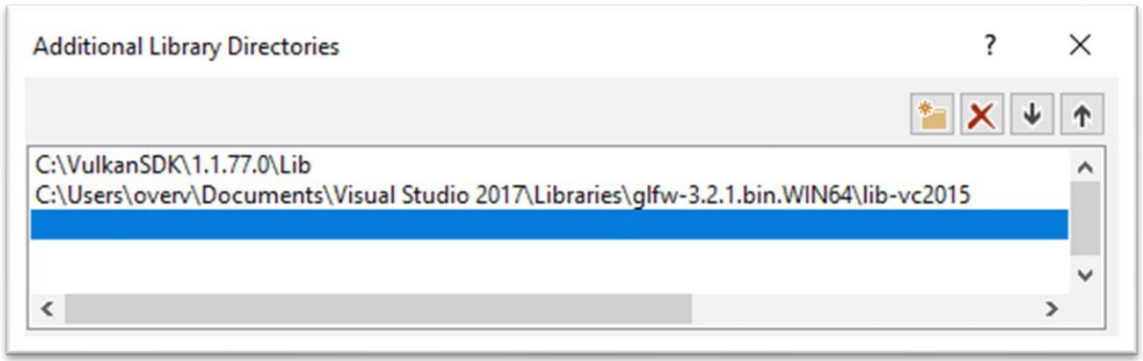
- U mapi gdje je instaliran Vulkan SDK nalazi se
  vkcude.exe

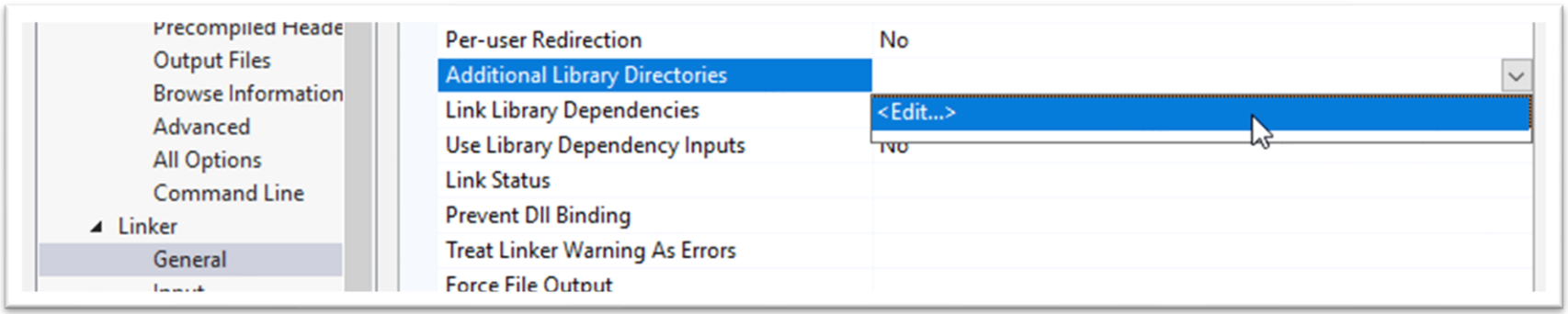
- Ukoliko je SDK instaliran pravilno i hardware
  podržava Vulkan, prikazati će se ovakva sličica
  kada se program pokrene

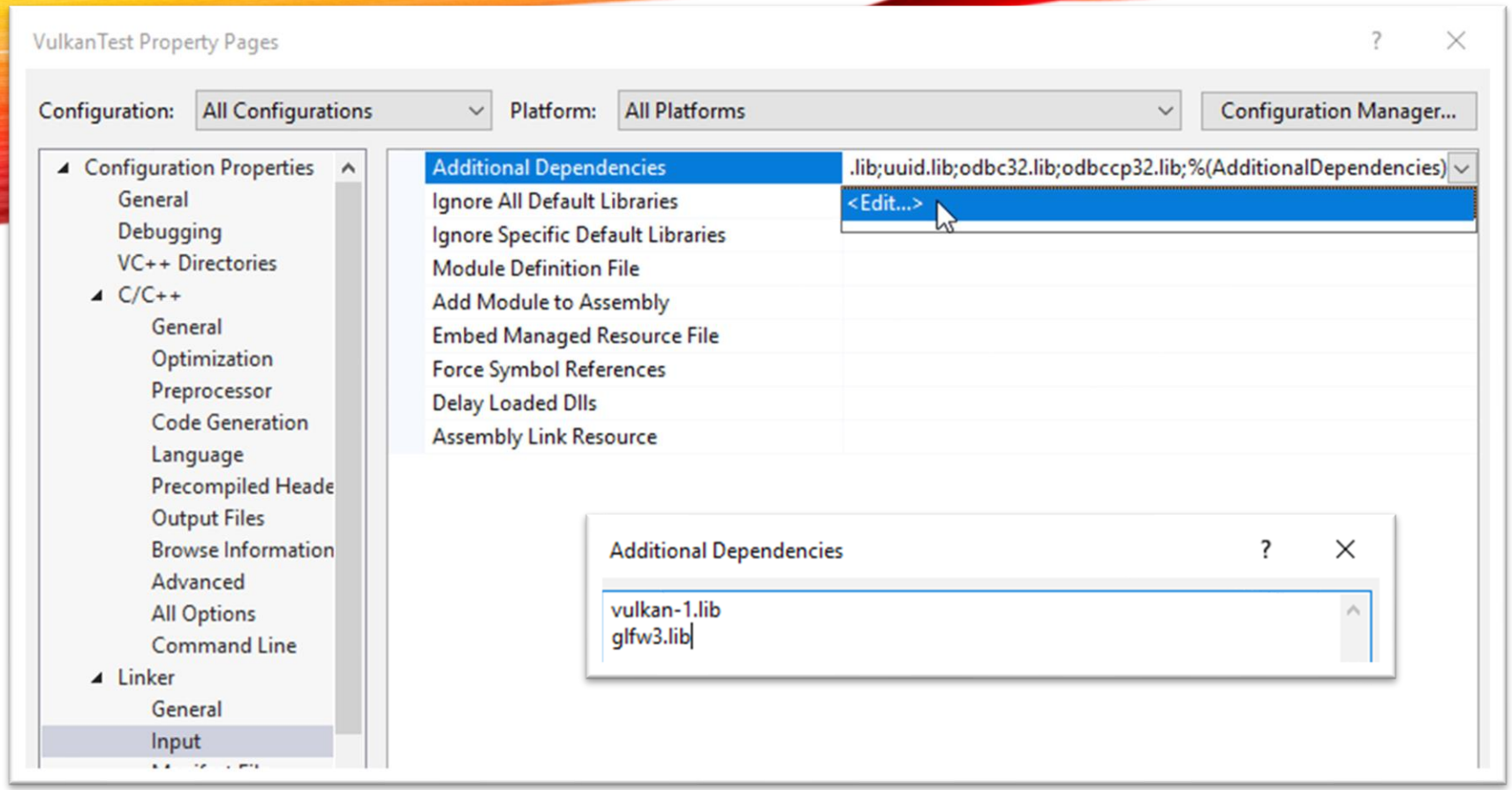


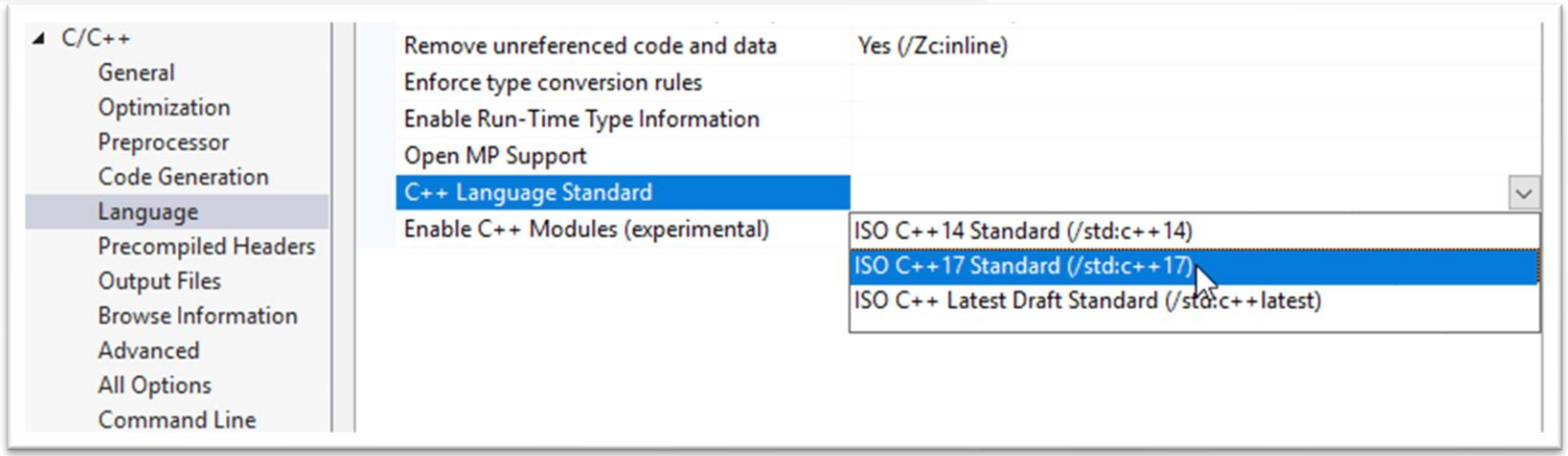

VulkanTest Property Pages
Configuration: All Configurations
Platform: All Platforms
Configuration Manager...
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Additional Include Directories
Additional #using Directories
Debug Information Format
Common Language RunTime Support
Consume Windows Runtime Extension
<Edit...>


Additional Include Directories
C:\VulkanSDK\1.1.77.0\Include
C:\Users\ \Documents\Visual Studio 2017\Libraries\glm
C:\Users\ \Documents\Visual Studio 2017\Libraries\glfw-3.2.1.bin.WIN64\include


Additional Library Directories
C:\VulkanSDK\1.1.77.0\Lib
C:\Users\overv\Documents\Visual Studio 2017\Libraries\glfw-3.2.1.bin.WIN64\lib-vc2015


Output Files
Browse Information
Advanced
All Options
Command Line
Linker
General
Per-user Redirection
No
Additional Library Directories
Link Library Dependencies
<Edit...>
Use Library Dependency Inputs
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output


VulkanTest Property Pages
Configuration: All Configurations
Platform: All Platforms
Configuration Manager...
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Heade
Output Files
Browse Information
Advanced
All Options
Command Line
Linker
General
Input
Additional Dependencies
.lib;uuid.lib;odbc32.lib;odbccp32.lib;%(AdditionalDependencies)
Ignore All Default Libraries
<Edit...>
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded Dlls
Assembly Link Resource
Additional Dependencies
vulkan-1.lib
glfw3.lib


C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
All Options
Command Line
Remove unreferenced code and data
Yes (/Zc:inline)
Enforce type conversion rules
Enable Run-Time Type Information
Open MP Support
C++ Language Standard
Enable C++ Modules (experimental)
ISO C++14 Standard (/std:c++14)
ISO C++17 Standard (/std:c++17)
ISO C++ Latest Draft Standard (/std:c++latest)

SHADERI

- Vulkan API donio je veliku promjenu u pisanju shadera
- Shaderi se pišu u GLSL (OpenGL Shading Language)
- Pomoću kompajlera kompajliraju se u *bytecode* naziva SPIR-V
- SPIR-V *bytecode* je lakši za procesuirati GPU
- Korištenjem ovog formata shadera moguće je izbjeći greške kod rada programa kao i mogućnost da kod bude odbačen od strane upravljačkog programa

PRIMJER SHADERA I POSTUPAK KOMPAJLIRANJA

```
#version 450
#extension GL_ARB_separate_shader_objects : enable

layout(location = 0) out vec3 fragColor;

vec2 positions[3] = vec2[](
    vec2(0.0, -0.5),
    vec2(0.5, 0.5),
    vec2(-0.5, 0.5)
);

vec3 colors[3] = vec3[](
    vec3(1.0, 0.0, 0.0),
    vec3(0.0, 1.0, 0.0),
    vec3(0.0, 0.0, 1.0)
);

void main() {
    gl_Position = vec4(positions[gl_VertexIndex], 0.0, 1.0);
    fragColor = colors[gl_VertexIndex];
}
```

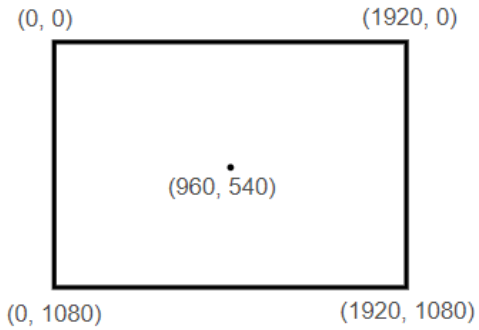
```
#version 450
#extension GL_ARB_separate_shader_objects : enable

layout(location = 0) in vec3 fragColor;

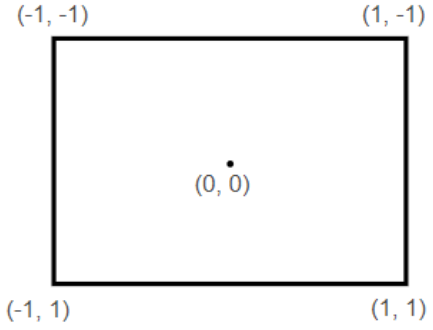
layout(location = 0) out vec4 outColor;

void main() {
    outColor = vec4(fragColor, 1.0);
}
```

Framebuffer coordinates



Normalized device coordinates



- Compile.bat

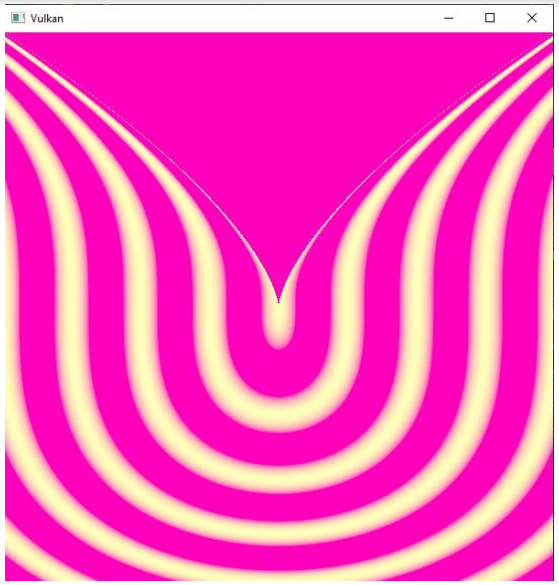
```
/home/user/VulkanSDK/x.x.x.x/x86_64/bin/glslc shader.vert -o vert.spv
/home/user/VulkanSDK/x.x.x.x/x86_64/bin/glslc shader.frag -o frag.spv
```



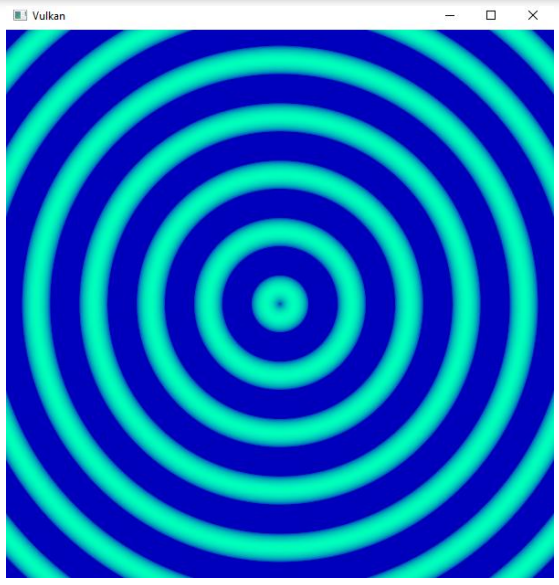
```
void initVulkan() {
    createInstance();
    setupDebugMessenger();
    createSurface();
    pickPhysicalDevice();
    createLogicalDevice();
    createSwapChain();
    createImageViews();
    createRenderPass();
    createGraphicsPipeline();
    createFramebuffers();
    createCommandPool();
    createVertexBuffer();
    createIndexBuffer();
    createCommandBuffers();
    createSyncObjects();
}
```

PRIMJERI

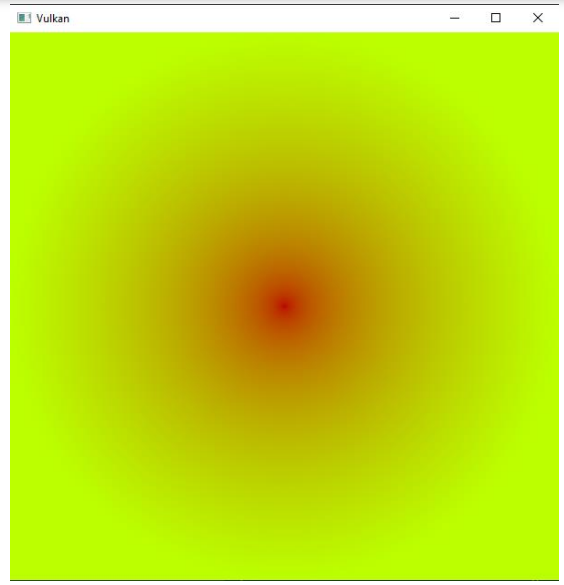
- Vulkan zahtjeva puno rada kod početnog postavljanja programa
- Primjer sa slike prikazuje korake postavljanja koji su izdvojeni kao funkcije




```
float r = sqrt(pow(v_tocka.x, 2.0) +
pow(v_tocka.y, 3.0));
bojaPiksela = vec4(1, cos(25.0 * r), 0.5, 1.0);
```




```
float r = sqrt(pow(v_tocka.x, 2.0) +
pow(v_tocka.y, 2.0));
bojaPiksela = vec4(0, sin(30.0 * r), 0.5, 1.0);
```




```
float r = sqrt(pow(v_tocka.x, 2.0) +
pow(v_tocka.y, 2.0));
bojaPiksela = vec4(0.5,  r, 0, 1.0);
```

PRIMJERI - NASTAVA

PRIMJERI - ZADAĆA

