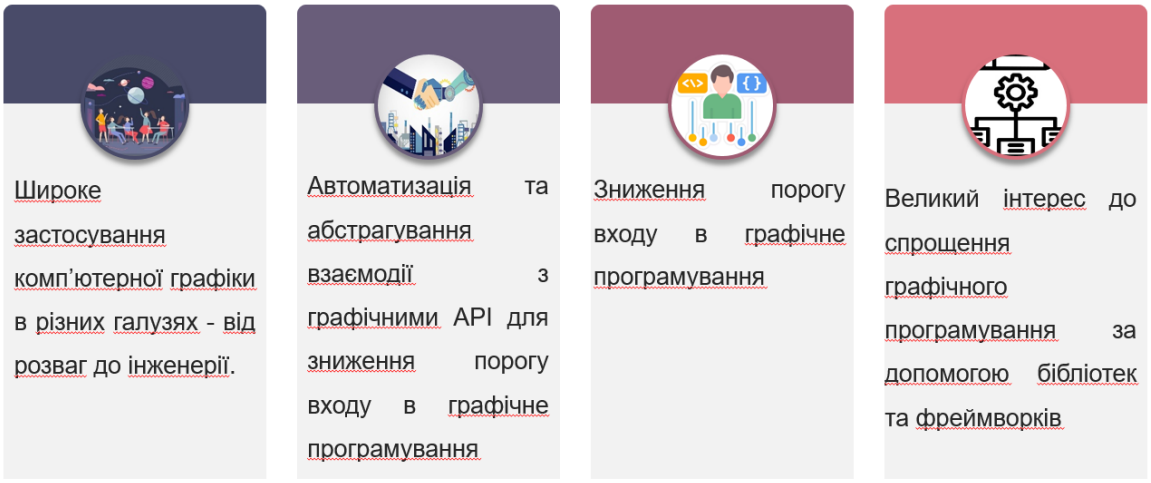
ДОДАТОК А

Графічний матеріал кваліфікаційної роботи



Існуючі рішення

Unity

Потужний кросплатформний ігровий рушій, який підтримує 2D та 3D графіку, фізику, анімацію та інтерактивність. Має зручний інтерфейс та велику кількість інструментів для розробки ігор і симуляцій, використовуючи мови C# або UnityScript.

SFML

Легка C++ бібліотека для створення мультимедійних застосунків, що надає простий доступ до графіки, звуку, вводу та віконної системи. Підходить для розробки 2D-ігор та візуалізацій.

BGFX

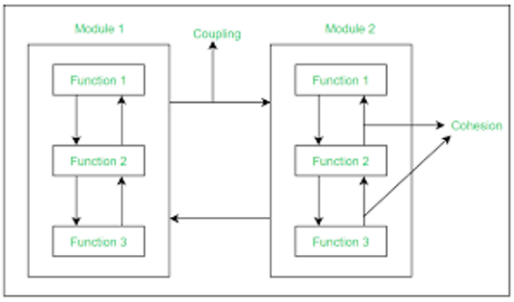
Низькорівнева кросплатформна бібліотека для рендерингу, яка підтримує кілька графічних API (OpenGL, DirectX, Vulkan, Metal). Вона орієнтована на розробників рушіїв і дає гнучкий контроль над графічним конвеєром.

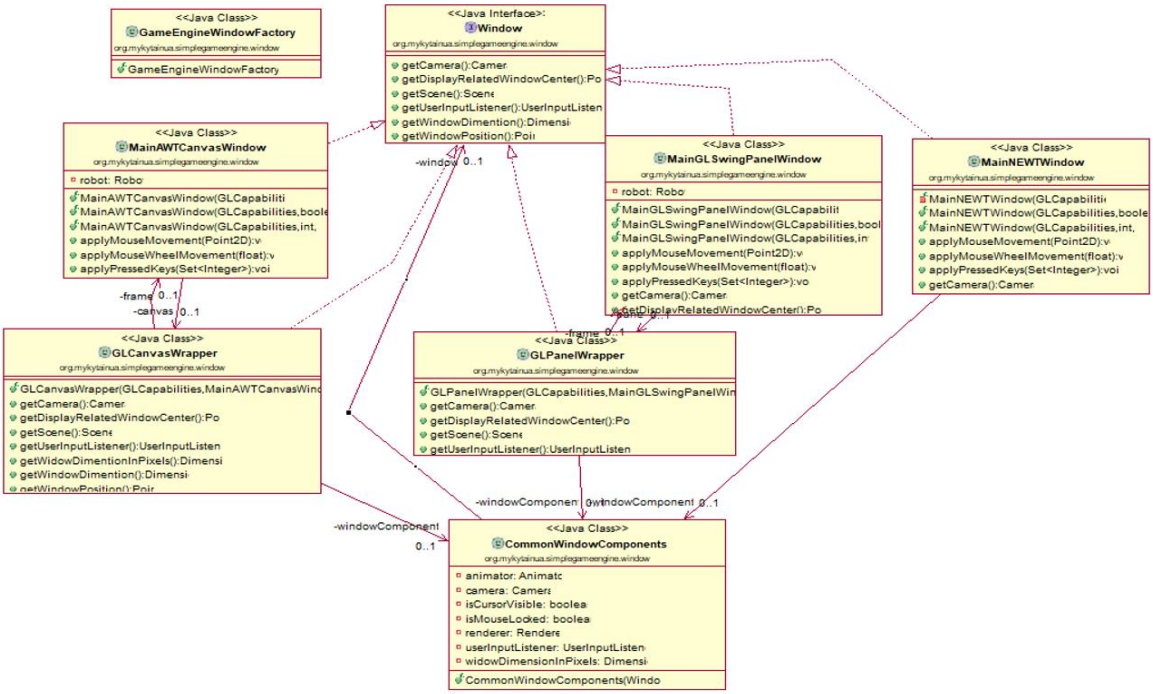
Використані технології

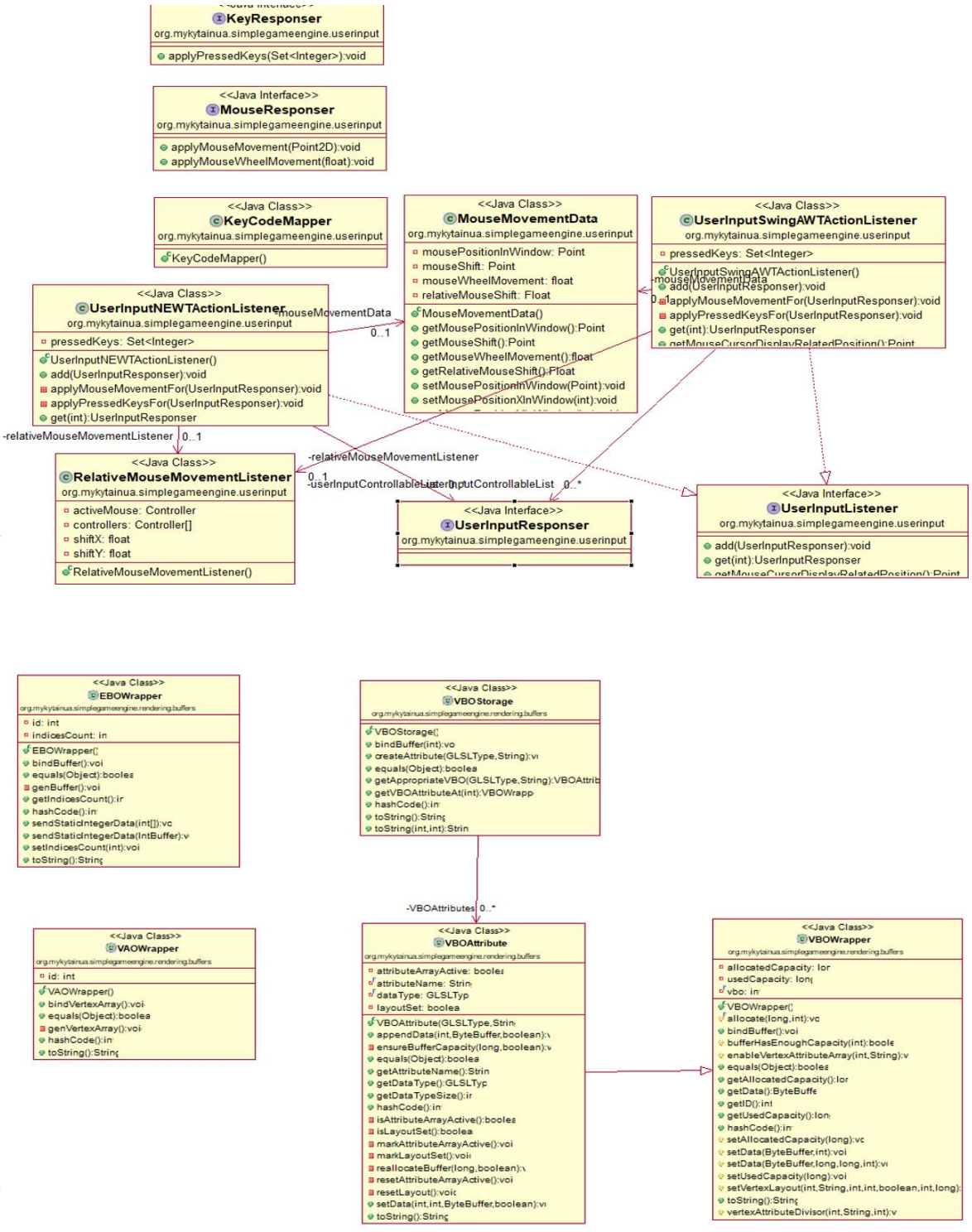
Java	об'єктно-орієнтована мова програмування, яка працює за принципом «написав один раз — запускай всюди».
Maven	система керування проєктами та залежностями в Java. Вона автоматизує процес збирання, тестування та підключення сторонніх бібліотек до проєкту.
OpenGL	кросплатформний графічний API для роботи з графікою. Він надає низькорівневий доступ до графічного процесора для створення візуальних ефектів.
Jinput	бібліотека на Java для роботи з пристроями введення. Вона забезпечує уніфікований доступ до контролерів на різних операційних системах.

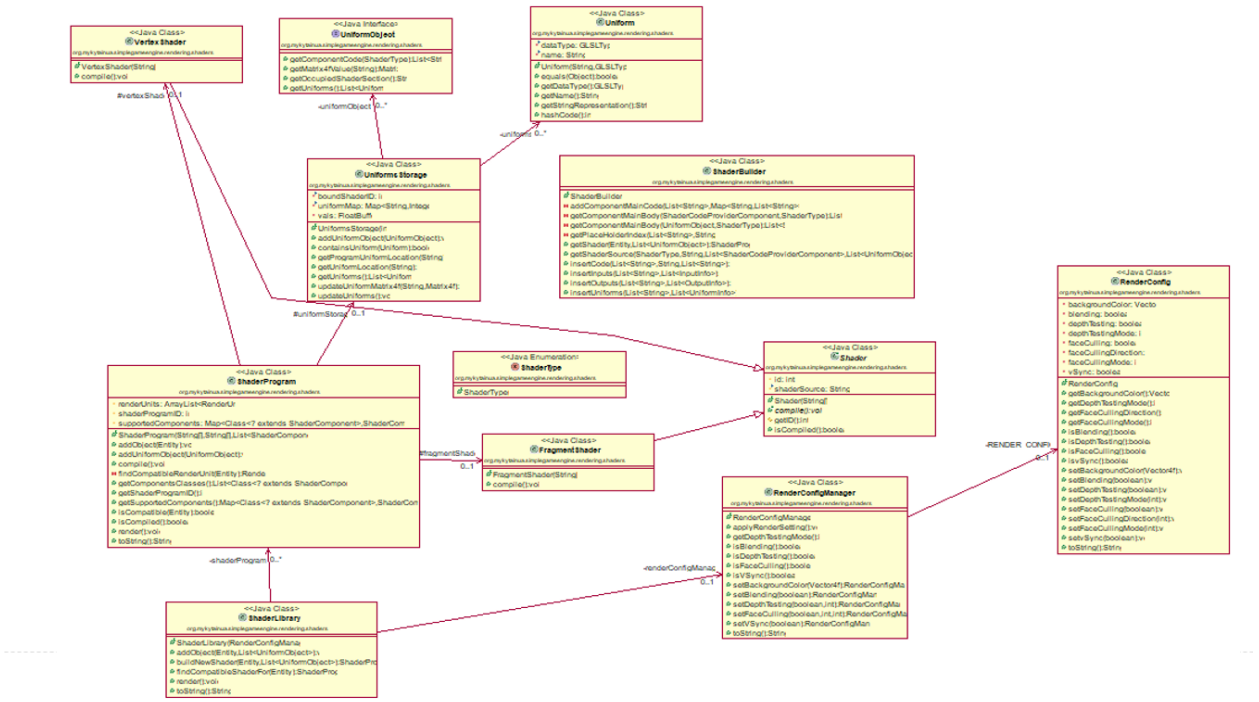
Постановка задачі

Потрібно створити бібліотеку, яка повинна забезпечити відображення 3D-об'єктів, обробку користувацького вводу та підтримку подальшого розширення (наприклад, фізики чи освітлення). Бібліотека має мати модулі для створення: вікон, рендеринг сцени, обробка вводу та консольний вивід. Загалом бібліотека має пмати такі характеристики як: продуктивність, кросплатформність та модульну архітектуру. Для реалізації було обрано мову Java, а також бібліотеки JOGL (для OpenGL), JInput (для вводу) та JOML (для спрощення математичних розрахунків).



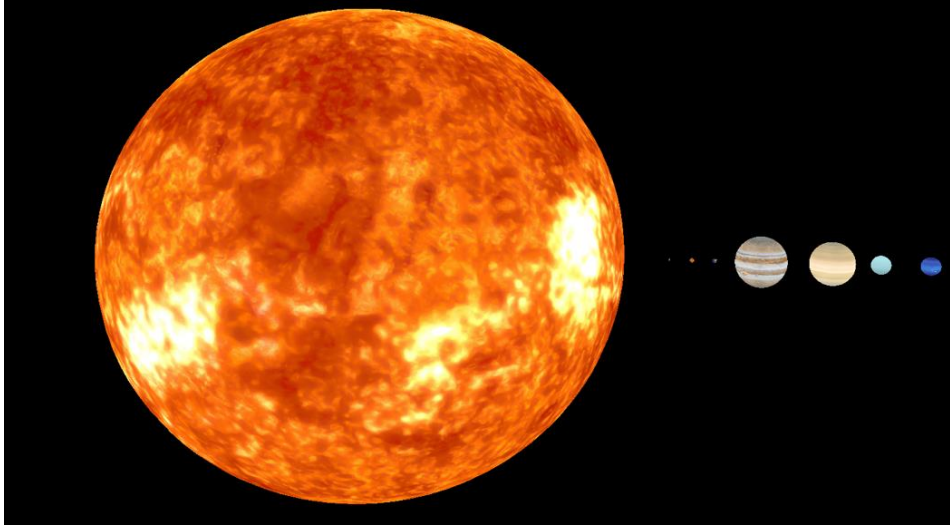



KeyResponser
org.mykytainua.simplegameengine.userinput
applyPressedKeys(Set<Integer>):void
<<Java Interface>>
MouseResponser
org.mykytainua.simplegameengine.userinput
applyMouseMovement(Point2D):void
applyMouseWheelMovement(float):void
<<Java Class>>
KeyCodeMapper
org.mykytainua.simplegameengine.userinput
KeyCodeMapper()
<<Java Class>>
MouseMovementData
org.mykytainua.simplegameengine.userinput
mousePositionInWindow: Point
mouseShift: Point
mouseWheelMovement: float
relativeMouseShift: Float
MouseMovementData()
getMousePositionInWindow():Point
getMouseShift():Point
getMouseWheelMovement():float
getRelativeMouseShift():Float
setMousePositionInWindow(Point):void
setMousePositionXInWindow(int):void
<<Java Class>>
UserInputSwingAWTActionListener
org.mykytainua.simplegameengine.userinput
pressedKeys: Set<Integer>
UserInputSwingAWTActionListener()
add(UserInputResponser):void
applyMouseMovementFor(UserInputResponser):void
applyPressedKeysFor(UserInputResponser):void
get(int):UserInputResponser
<<Java Class>>
UserInputNEWTActionListener
org.mykytainua.simplegameengine.userinput
pressedKeys: Set<Integer>
UserInputNEWTActionListener()
add(UserInputResponser):void
applyMouseMovementFor(UserInputResponser):void
applyPressedKeysFor(UserInputResponser):void
get(int):UserInputResponser
mouseMovementData
0..1
-relativeMouseMovementListener
0..1
<<Java Class>>
RelativeMouseMovementListener
org.mykytainua.simplegameengine.userinput
activeMouse: Controller
controllers: Controller[]
shiftX: float
shiftY: float
RelativeMouseMovementListener()
-relativeMouseMovementListener
0..1
-userInputControllableList
0..*
<<Java Interface>>
UserInputResponser
org.mykytainua.simplegameengine.userinput
<<Java Interface>>
UserInputListener
org.mykytainua.simplegameengine.userinput
add(UserInputResponser):void
get(int):UserInputResponser
<<Java Class>>
EBOWrapper
org.mykytainua.simplegameengine.rendering.buffers
id: int
indicesCount: in
EBOWrapper()
bindBuffer():voi
equals(Object):boolea
genBuffer():voi
getIndicesCount():ir
hashCode():in
sendStaticIntegerData(int[]):vo
sendStaticIntegerData(IntBuffer):v
setIndicesCount(int):voi
toString():String
<<Java Class>>
VBOStorage
org.mykytainua.simplegameengine.rendering.buffers
VBOStorage()
bindBuffer(int):vo
createAttribute(GLSLType,String):vo
equals(Object):boolea
getAppropriateVBO(GLSLType,String):VBOAttrib
getVBOAttributeAt(int):VBOWrapp
hashCode():in
toString():String
toString(int,int):Strin
-VBOAttributes
0..*
<<Java Class>>
VAOWrapper
org.mykytainua.simplegameengine.rendering.buffers
id: int
VAOWrapper()
bindVertexArray():voi
equals(Object):boolea
genVertexArray():voi
hashCode():in
toString():String
<<Java Class>>
VBOAttribute
org.mykytainua.simplegameengine.rendering.buffers
attributeArrayActive: boolea
attributeName: Strin
dataType: GLSLTyp
layoutSet: boolea
VBOAttribute(GLSLType,Strin
appendData(int,ByteBuffer,boolean):v
ensureBufferCapacity(long,boolean):v
equals(Object):boolea
getAttributeName():Strin
getDataType():GLSLTyp
getDataTypeSize():ir
hashCode():in
isAttributeArrayActive():boolea
isLayoutSet():boolea
markAttributeArrayActive():voi
markLayoutSet():voi
reallocateBuffer(long,boolean):v
resetAttributeArrayActive():voi
resetLayout():voi
setData(int,int,ByteBuffer,boolean):vo
toString():String
<<Java Class>>
VBOWrapper
org.mykytainua.simplegameengine.rendering.buffers
allocatedCapacity: lor
usedCapacity: lon
vbo: in
VBOWrapper()
allocate(long,int):vo
bindBuffer():voi
bufferHasEnoughCapacity(int):boole
enableVertexAttributeArray(int,String):v
equals(Object):boolea
getAllocatedCapacity():lor
getData():ByteBuffe
getID():int
getUsedCapacity():lon
hashCode():in
setAllocatedCapacity(long):vo
setData(ByteBuffer,int):voi
setData(ByteBuffer,long,long,int):vo
setUsedCapacity(long):voi
setVertexLayout(int,String,int,int,boolean,int,long):
toString():String
vertexAttributeDivisor(int,String,int):v


Vertex Shader
UniformObject
Uniform
Uniforms Storage
ShaderBuilder
RenderConfig
Shader
ShaderType
ShaderProgram
FragmentShader
RenderConfigManager
ShaderLibrary

Демонстрація можливостей: молекула води

Демонстрація можливостей: сонячна система



Демонстрація можливостей: кімната

Висновки

У ході роботи була створена кросплатформна бібліотека для 3D-візуалізації на Java з використанням OpenGL через обгортку JOGL. Для обробки користувацького вводу — JInput, що дозволило уникнути обмежень стандартних засобів Java. Архітектура бібліотеки побудована на компонентній моделі (ECS), яка забезпечує високу гнучкість, масштабованість та можливість автоматичної генерації шейдерів. На відміну від складніших рішень, таких як Java3D чи LWJGL, ця бібліотека орієнтована на простоту та навчальні потреби. У перспективі вона може бути розширена підтримкою фізики, освітлення, GUI та сучасних графічних API, таких як Vulkan. Проєкт продемонстрував практичну цінність і дав цінний досвід у сфері графічного програмування.