ДОДАТОК А

Приклади програмної реалізації основного функціоналу

```python
bl_info = {
    "name": "AI PBR Generator (Seamless High-Quality)",
    "author": "Your Name",
    "version": (1, 5),
    "blender": (3, 0, 0),
    "location": "Shader Editor > Sidebar > AI PBR",
    "description": "Generates seamless PBR maps using high-overlap tiled inference",
    "category": "Node",
}

import bpy
import os
import sys
import numpy as np
import math


current_dir = os.path.dirname(os.path.realpath(__file__))
modules_path = os.path.join(current_dir, "modules")

if modules_path not in sys.path:
    sys.path.append(modules_path)

try:
    import cv2
except ImportError:
    cv2 = None
    print("AI PBR Error: OpenCV not found. Please install into 'modules' folder.")
```

Рисунок А1 – Модуль ініціалізації та налаштування оточення

```python
def postprocess_smart(tensor):
    """Handles NCHW/NHWC and -1..1 -> 0..1 conversion."""
    arr = tensor[0]
    shape = arr.shape

    if shape[0] in [1, 3] and (len(shape) == 3 and shape[1] > 3):
        arr = np.transpose(arr, (1, 2, 0))
    elif shape[-1] in [1, 3] and (len(shape) == 3 and shape[0] > 3):
        pass

    arr = (arr + 1.0) / 2.0
    arr = np.clip(arr, 0.0, 1.0)
    return arr
```

Рисунок А2 – Функція нормалізації та транспонування вихідних тензорів

```python
def infer_single_tile(net, tile_rgb):
    """Runs inference on a single chunk."""
    norm_img = (tile_rgb * 2.0) - 1.0
    input_blob = np.expand_dims(norm_img, axis=0)

    net.setInput(input_blob)
    out_names = net.getUnconnectedOutLayersNames()
    outputs = net.forward(out_names)

    pred_norm = None
    pred_rough = None

    for out in outputs:
        shape = out.shape
        check_shape = out[0].shape
        c_nchw = check_shape[0]
        c_nhwc = check_shape[-1]

        is_normal = (c_nchw == 3 or c_nhwc == 3)
        is_rough = (c_nchw == 1 or c_nhwc == 1)

        if is_normal:
            pred_norm = postprocess_smart(out)
        elif is_rough:
            pred_rough = postprocess_smart(out)

    if pred_rough is not None and len(pred_rough.shape) == 3:
        pred_rough = pred_rough[:, :, 0]

    return pred_norm, pred_rough
```

Рисунок А3 – Функція інференсу для окремого фрагмента зображення

```python
def process_image_seamless_hq(image_node, model_path):
    if not cv2:
        raise ImportError("OpenCV module not loaded.")

    src_image = image_node.image
    if not src_image: return {"ERROR": "No image loaded."}

    width, height = src_image.size
    print(f"AI PBR: Processing Image {width}x{height} with High Overlap")

    pixels = np.array(src_image.pixels[:], dtype=np.float32)
    pixels = pixels.reshape((height, width, 4))
    img_rgb = pixels[:, :, :3]

    net = cv2.dnn.readNetFromONNX(model_path)
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

    MODEL_INPUT_SIZE = 256
    VALID_SIZE = 192
    BORDER = (MODEL_INPUT_SIZE - VALID_SIZE) // 2

    full_normal = np.zeros((height, width, 3), dtype=np.float32)
    full_roughness = np.zeros((height, width), dtype=np.float32)

    cols = math.ceil(width / VALID_SIZE)
    rows = math.ceil(height / VALID_SIZE)

    print(f"AI PBR: HQ Seamless Grid is {cols} x {rows}")

    img_padded = cv2.copyMakeBorder(img_rgb, BORDER, BORDER, BORDER, BORDER, cv2.BORDER_REFLECT)
    pad_h, pad_w, _ = img_padded.shape

    for y in range(rows):
        for x in range(cols):
            cx = x * VALID_SIZE
            cy = y * VALID_SIZE

            x_start_pad = cx
            y_start_pad = cy

            x_start_pad = min(x_start_pad, pad_w - MODEL_INPUT_SIZE)
            y_start_pad = min(y_start_pad, pad_h - MODEL_INPUT_SIZE)

            x_end_pad = x_start_pad + MODEL_INPUT_SIZE
            y_end_pad = y_start_pad + MODEL_INPUT_SIZE

            chunk = img_padded[y_start_pad:y_end_pad, x_start_pad:x_end_pad]

            tile_norm, tile_rough = infer_single_tile(net, chunk)

            valid_norm = tile_norm[BORDER:BORDER+VALID_SIZE, BORDER:BORDER+VALID_SIZE]
            valid_rough = tile_rough[BORDER:BORDER+VALID_SIZE, BORDER:BORDER+VALID_SIZE]

            final_x_start = cx
            final_y_start = cy
            final_x_end = min(final_x_start + VALID_SIZE, width)
            final_y_end = min(final_y_start + VALID_SIZE, height)

            paste_w = final_x_end - final_x_start
            paste_h = final_y_end - final_y_start

            paste_norm = valid_norm[0:paste_h, 0:paste_w]
            paste_rough = valid_rough[0:paste_h, 0:paste_w]

            full_normal[final_y_start:final_y_end, final_x_start:final_x_end] = paste_norm
            full_roughness[final_y_start:final_y_end, final_x_start:final_x_end] = paste_rough

    print("AI PBR: Stitching complete. Packing image...")
    alpha = np.ones((height, width, 1), dtype=np.float32)

    if full_normal.shape[-1] != 3:
        full_normal = np.transpose(full_normal, (1, 2, 0))
    normal_rgba = np.dstack((full_normal, alpha))

    rough_rgb = np.dstack((full_roughness, full_roughness, full_roughness))
    rough_rgba = np.dstack((rough_rgb, alpha))

    return normal_rgba, rough_rgba
```

Рисунок А4 – Реалізація алгоритму безшовної генерації методом тайлінгу

```python
class PBR_OT_GenerateSeamlessHQ(bpy.types.Operator):
    """Generate Seamless Hi-Res Maps (HQ)"""
    bl_idname = "node.generate_pbr_seamless_hq"
    bl_label = "Generate PBR Maps (HQ)"

    def execute(self, context):
        try:
            active_node = context.active_node
            if not active_node or active_node.type != 'TEX_IMAGE':
                self.report({'ERROR'}, "Select an Image Node.")
                return {'CANCELLED'}
        except:
            return {'CANCELLED'}

        model_file = os.path.join(current_dir, "pbr_unet_v2.onnx")

        try:
            self.report({'INFO'}, "Processing HQ Seamless Tiling...")
            norm_px, rough_px = process_image_seamless_hq(active_node, model_file)
        except Exception as e:
            self.report({'ERROR'}, f"Error: {e}")
            import traceback
            traceback.print_exc()
            return {'CANCELLED'}

        base_name = active_node.image.name.split('.')[0]
        norm_img = create_blender_image(f"{base_name}_NRM", norm_px)
        rough_img = create_blender_image(f"{base_name}_RGH", rough_px)

        # Connect Nodes
        tree = context.space_data.node_tree
        nodes = tree.nodes

        # Normal Node
        n_node = nodes.new('ShaderNodeTexImage')
        n_node.image = norm_img
        n_node.label = "AI Normal"
        n_node.location = (active_node.location.x + 300, active_node.location.y)
        n_node.image.colorspace_settings.name = 'Non-Color'

        # Roughness Node
        r_node = nodes.new('ShaderNodeTexImage')
        r_node.image = rough_img
        r_node.label = "AI Roughness"
        r_node.location = (active_node.location.x + 300, active_node.location.y - 300)
        r_node.image.colorspace_settings.name = 'Non-Color'

        return {'FINISHED'}
```

Рисунок А5 – Оператор генерації та інтеграція з Blender API

```python
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D, concatenate, LeakyReLU, BatchNormalization, Dropout
from tensorflow.keras.models import Model

def build_pbr_unet_stable(input_shape=(256, 256, 3)):
    inputs = Input(input_shape)

    c1 = Conv2D(64, 3, activation='relu', padding='same')(inputs)
    c1 = Conv2D(64, 3, activation='relu', padding='same')(c1)
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = Conv2D(128, 3, activation='relu', padding='same')(p1)
    c2 = Conv2D(128, 3, activation='relu', padding='same')(c2)
    p2 = MaxPooling2D((2, 2))(c2)

    c3 = Conv2D(256, 3, activation='relu', padding='same')(p2)
    c3 = Conv2D(256, 3, activation='relu', padding='same')(c3)
    p3 = MaxPooling2D((2, 2))(c3)

    c4 = Conv2D(512, 3, activation='relu', padding='same')(p3)
    c4 = Conv2D(512, 3, activation='relu', padding='same')(c4)
    p4 = MaxPooling2D((2, 2))(c4)

    c5 = Conv2D(1024, 3, activation='relu', padding='same')(p4)
    c5 = Conv2D(1024, 3, activation='relu', padding='same')(c5)

    u6 = UpSampling2D((2, 2))(c5)
    u6 = Conv2D(512, 2, activation='relu', padding='same')(u6)
    u6 = concatenate([u6, c4])
    c6 = Conv2D(512, 3, activation='relu', padding='same')(u6)
    c6 = Conv2D(512, 3, activation='relu', padding='same')(c6)

    u7 = UpSampling2D((2, 2))(c6)
    u7 = Conv2D(256, 2, activation='relu', padding='same')(u7)
    u7 = concatenate([u7, c3])
    c7 = Conv2D(256, 3, activation='relu', padding='same')(u7)
    c7 = Conv2D(256, 3, activation='relu', padding='same')(c7)

    u8 = UpSampling2D((2, 2))(c7)
    u8 = Conv2D(128, 2, activation='relu', padding='same')(u8)
    u8 = concatenate([u8, c2])
    c8 = Conv2D(128, 3, activation='relu', padding='same')(u8)
    c8 = Conv2D(128, 3, activation='relu', padding='same')(c8)

    u9 = UpSampling2D((2, 2))(c8)
    u9 = Conv2D(64, 2, activation='relu', padding='same')(u9)
    u9 = concatenate([u9, c1])
    c9 = Conv2D(64, 3, activation='relu', padding='same')(u9)
    c9 = Conv2D(64, 3, activation='relu', padding='same')(c9)

    out_normal = Conv2D(3, 1, activation='tanh', name='normal')(c9)
    out_roughness = Conv2D(1, 1, activation='tanh', name='roughness')(c9)

    model = Model(inputs=[inputs], outputs=[out_normal, out_roughness])
    return model

model = build_pbr_unet_stable()
```

Рисунок А6 – Приклад U-Net моделі

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	93 с.
	Інші документи	
2.	Презентаційні матеріали	18 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Ніколайчук Д.А.		09.12	Генерація PBR-текстур на основі зображень у 3D-графіці методами машинного навчання	Шифр групи	Код напр./спец.
Перевірила		Вітько О. В		10.12		ДСм-24-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					