

Міністерство освіти і науки України  
Національний університет «Одеська політехніка»  
Інститут комп’ютерних систем  
Кафедра інформаційних систем

Сучков Олександр Ігорович

студент групи АІ-221

## **ПРОЄКТНА РОБОТА**

З дисципліни «Технології створення ПП»

Розробка програми управління відеогрою за допомогою тіла користувача  
через камеру

Спеціальність: 122 Комп’ютерні науки

Освітня програма: Комп’ютерні науки

Керівник: доц. Блажко Олександр Анатолійович

Одеса – 2024

**Мета:** Розробка програмного забезпечення, що дає змогу керувати відеогрою за допомогою рухів тіла користувача, розпізнаваних через камеру, з використанням бібліотеки MediaPipe і емуляції клавіш клавіатури та стіків геймпада.

**Використані технології:** Python 3.11, OpenCV, MediaPipe, pyupnp, vgamepad (XInput), Win32 API, Numpy.

Програма призначена для управління відеоіграми за допомогою жестів тіла, розпізнаваних через звичайну вебкамеру. Вона реалізує повноцінну емуляцію геймпадного вводу (ліва/права рука як стіки), а також клавіш миші й клавіатури на основі положення частин тіла користувача. Вся логіка візуалізується у вигляді оверлею на відео з камери. Через відкритий код програми, вона може бути вільно модифікована або доповнена для більш широкого функціоналу.

### **Функціонал програми:**

#### *1. Відеопотік та розпізнавання тіла*

- Здійснюється через бібліотеку *MediaPipe*, яка в реальному часі обчислює 3D-координати основних точок тіла (нос, плечі, руки, коліна, ступні).
- Користувач бачить себе в режимі реального часу разом зі скелетом і графічним оверлеєм, що відображає дії.

#### *2. Емуляція лівого стіку геймпада (ліва рука)*

- Рух лівої руки відносно плеча (через віддзеркаленість камери у коді використовується права рука) використовується для керування *лівим аналоговим стіком геймпада*, який відповідає за рух персонажа в більшості ігор.
- Рух уперед/назад/вліво/вправо передається через бібліотеку *vgamepad* у форматі *XInput* з обмеженням чутливості та порогом, щоб уникнути випадкових коливань.

#### *3. Емуляція правого стіку геймпада (права рука)*

- *Права рука*, аналогічно до емуляції лівого стіку, використовується для керування *правим стіком*, який відповідає за обертання камери або приціл у більшості сучасних ігор від першого або третього обличчя.
- Також реалізовано масштабування чутливості, щоб дозволити гнучке керування при невеликих жестах.

#### *4. Керування мишкою: ЛКМ та ПКМ*

- Клацання лівої або правої кнопки миші ініціюється *нахилом голови* користувача вліво або вправо.
- Визначення повороту виконується шляхом аналізу кута між носом та центром плечей.
- Натискання й відпускання миші реалізовано через *pyupnp.mouse*.

## 5. Емуляція клавіші *Shift*

- Коли ліве коліно піднімається вище лівого стегна, активується натискання клавіші *Shift*.
- Така поза моделює біг або стрибок, і дозволяє виконувати натискання *Shift* незалежно від рухів руками та головою.

## 6. Емуляція клавіші *Space*

- Якщо ліва ступня заходить лівіше за праве коліно, програма інтерпретує це як жест для натискання *Space*.
- Ця поза може бути суміщена з позою *Shift* для одночасного або швидкого послідовного натискання, і так само може бути виконана незалежно від рук та голови.

## 7. Візуалізація — графічний оверлей

- На відео з камери виводиться:
  - LStick і RStick у вигляді двох кругів з точками, що показують напрям руху.
  - Індикатори натискань LMB, RMB, *Shift*, *Space* у вигляді кольорових кнопок у верхній частині екрана.
  - Скелет тіла для зворотного зв'язку — візуально видно, які точки розпізнає модель.

## 8. Інтерфейс користувача

- Камера запускається в окремому вікні з роздільною здатністю 640×480, яке автоматично виводиться поверх інших вікон.

Цей функціонал робить програму універсальним інструментом для експериментального управління іграми через тіло без додаткового обладнання для ігрозації фізичних вправ. Програма підтримує будь-яку гру, яка сприймає XInput (геймпад Xbox) або звичайні клавіші клавіатури й миші.

### Код програми:

```
import cv2
import mediapipe as mp
import time
from pynput.keyboard import Controller, Key
from pynput.mouse import Controller as MouseController, Button
mouse = MouseController()
import numpy as np
import vgamerpad as vg # pip install vgamerpad
import win32gui
import win32con

# Ініціалізація Mediapipe
mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils
pose = mp_pose.Pose(min_detection_confidence=0.7, min_tracking_confidence=0.7)

# Ініціалізація для керування клавіатурою та віртуальним геймпадом
keyboard = Controller()
gamepad = vg.VX360Gamepad()

# Налаштування відображення
show_skeleton = True
```

```

# Змінні для відстеження стану
head_left = False
head_right = False
right_knee = False
stick_dx = 0.0
stick_dy = 0.0

# Параметри порогів
WASD_THRESHOLD = 0.05          # значний нахил для WASD
RSTICK_THRESHOLD = 0.05         # значний нахил для правого стіку
HEAD_ANGLE_THRESHOLD = 15       # градусів для ЛКМ/ПКМ
KNEE_THRESHOLD = 0.0            # підняття коліна

# Стан клавіш WASD
wasd_state = {'w': False, 'a': False, 's': False, 'd': False}

# Функція для встановлення вікна поверх інших
def set_window_topmost(name):
    hwnd = win32gui.FindWindow(None, name)
    if hwnd:
        win32gui.SetWindowPos(hwnd, win32con.HWND_TOPMOST, 0, 0, 0, 0,
                              win32con.SWP_NOMOVE | win32con.SWP_NOSIZE)

# Емуляція лівого стіку за допомогою правої руки
left_dx = 0.0
left_dy = 0.0

def emulate_wasd(landmarks):
    global wasd_state # {'w': False, 'a': False, 's': False, 'd': False}
    rs = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER]
    rw = landmarks[mp_pose.PoseLandmark.RIGHT_WRIST]
    dx = (rw.x - rs.x) * 6
    dy = (rs.y - rw.y) * 6 # інвертована вісь Y

    # Застосовуємо поріг
    if abs(dx) < WASD_THRESHOLD:
        dx = 0.0
    if abs(dy) < WASD_THRESHOLD:
        dy = 0.0

    def to_axis(v):
        return int(max(-1.0, min(1.0, v)) * 32767)

    gamepad.left_joystick(x_value=to_axis(dx), y_value=to_axis(dy))
    gamepad.update()

    global left_dx, left_dy
    left_dx, left_dy = dx, dy

    # Для оверлею – умовні WASD
    active = []
    if dy > WASD_THRESHOLD: active.append('w')
    if dy < -WASD_THRESHOLD: active.append('s')
    if dx < -WASD_THRESHOLD: active.append('a')
    if dx > WASD_THRESHOLD: active.append('d')
    return active

# Емуляція правого стіку Xbox через vgamerpad

def emulate_rstick(landmarks):
    global stick_dx, stick_dy
    ls = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER]
    lw = landmarks[mp_pose.PoseLandmark.LEFT_WRIST]
    dx = (lw.x - ls.x)*4
    dy = (ls.y - lw.y)*4 # інвертуємо Y для осі стіку

    # Застосовуємо поріг

```

```

if abs(dx) < RSTICK_THRESHOLD:
    dx = 0.0
if abs(dy) < RSTICK_THRESHOLD:
    dy = 0.0

stick_dx, stick_dy = dx, dy

# Функція нормалізації в діапазон -32768..32767
def to_axis(v):
    return int(max(-1.0, min(1.0, v)) * 32767)

x_val = to_axis(dx)
y_val = to_axis(dy)

gamepad.right_joystick(x_value=x_val, y_value=y_val)
gamepad.update()

# Перевірка нахилу голови на кут для кліків

def check_head(landmarks):
    global head_left, head_right
    nose = landmarks[mp_pose.PoseLandmark.NOSE]
    ls = landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER]
    rs = landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER]

    cx = (ls.x + rs.x) / 2
    cy = (ls.y + rs.y) / 2

    angle = np.degrees(np.arctan2(nose.x - cx, cy - nose.y))
    left_click = False
    right_click = False

    if angle > HEAD_ANGLE_THRESHOLD and not head_left:
        head_left = True
        left_click = True
        mouse.press(Button.left)
    elif angle <= HEAD_ANGLE_THRESHOLD and head_left:
        head_left = False
        mouse.release(Button.left)

    if angle < -HEAD_ANGLE_THRESHOLD and not head_right:
        head_right = True
        right_click = True
        mouse.press(Button.right)
    elif angle >= -HEAD_ANGLE_THRESHOLD and head_right:
        head_right = False
        mouse.release(Button.right)

    return head_left, head_right

# Перевірка підніняття коліна для Shift

# Стани для Shift і Space
shift_state = False
space_state = False

def check_knee(landmarks):
    global shift_state, space_state

    # SHIFT – ліве коліно вище за лівий таз
    lh = landmarks[mp_pose.PoseLandmark.LEFT_HIP]
    lk = landmarks[mp_pose.PoseLandmark.LEFT_KNEE]

    if lk.y < lh.y - KNEE_THRESHOLD and not shift_state:
        shift_state = True
        keyboard.press(Key.shift)
    elif lk.y >= lh.y - KNEE_THRESHOLD and shift_state:
        shift_state = False
        keyboard.release(Key.shift)

```

```

# SPACE – якщо ліва ступня правіше за праве коліно
lf = landmarks[mp_pose.PoseLandmark.LEFT_FOOT_INDEX]
rk = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE]

if lf.x <= rk.x and not space_state:
    space_state = True
    keyboard.press(Key.space)
elif lf.x > rk.x and space_state:
    space_state = False
    keyboard.release(Key.space)

return shift_state, space_state

# Функція для малювання оверлею

def draw_overlay(frame, left_dx, left_dy, stick_dx, stick_dy, head_left, head_right,
shift_state, space_state):
    h, w, _ = frame.shape
    size = 50
    margin = 10
    ox = w - (size*3 + margin*4)
    oy = h - (size*3 + margin*4)

    # Малюємо лівий стік зліва внизу
    cx_l = margin + size
    cy_l = h - margin - size
    cv2.circle(frame, (cx_l, cy_l), size, (180, 180, 180), 2)
    dot_x_l = int(cx_l + left_dx * size * 2)
    dot_y_l = int(cy_l - left_dy * size * 2)
    cv2.circle(frame, (dot_x_l, dot_y_l), 8, (0, 255, 0), -1)
    cv2.putText(frame, 'LStick', (cx_l - size, cy_l - size - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

    # Малюємо правий стік як коло
    cx, cy = w - margin - size, h - size - margin
    cv2.circle(frame, (cx, cy), size, (180, 180, 180), 2)
    dot_x = int(cx + stick_dx * size * 2)
    dot_y = int(cy - stick_dy * size * 2)
    cv2.circle(frame, (dot_x, dot_y), 8, (0, 255, 0), -1)
    cv2.putText(frame, 'RStick', (cx - size, cy - size - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (255, 255, 255), 1)

    # Клики + Shift + Space
    labels = [('LMB', head_left), ('RMB', head_right), ('Shift', shift_state), ('Space',
space_state)]
    for idx, (label, pressed) in enumerate(labels):
        x = margin + idx * (size * 2)
        y = margin
        color = (0, 255, 0) if pressed else (100, 100, 100)
        cv2.rectangle(frame, (x, y), (x + size * 2, y + size), color, -1)
        cv2.putText(frame, label, (x + 10, y + 35), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2)

# Головний цикл

def main():
    cap = cv2.VideoCapture(0)
    cv2.namedWindow('Controller', cv2.WINDOW_NORMAL)
    cv2.resizeWindow('Controller', 640, 480)
    set_window_topmost('Controller')

    while True:
        ret, frame = cap.read()
        if not ret: break
        frame = cv2.flip(frame, 1)
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

```

```

res = pose.process(rgb)
wasd_active = []
if res.pose_landmarks:
    lm = res.pose_landmarks.landmark
    if show_skeleton:
        mp_drawing.draw_landmarks(frame, res.pose_landmarks,
mp_pose.POSE_CONNECTIONS)
        wasd_active = emulate_wasd(lm)
        emulate_rstick(lm)
        hl, hr = check_head(lm)
        kn = check_knee(lm)
        draw_overlay(frame, left_dx, left_dy, stick_dx, stick_dy, head_left, head_right,
shift_state, space_state)
        cv2.imshow('Controller', frame)
        if cv2.waitKey(1) & 0xFF in (27, ord('q')): break

# Відпускаємо всі натиснуті клавіші
for k, pressed in wasd_state.items():
    if pressed: keyboard.release(k)
if head_left or head_right or right_knee:
    keyboard.release(Key.shift)

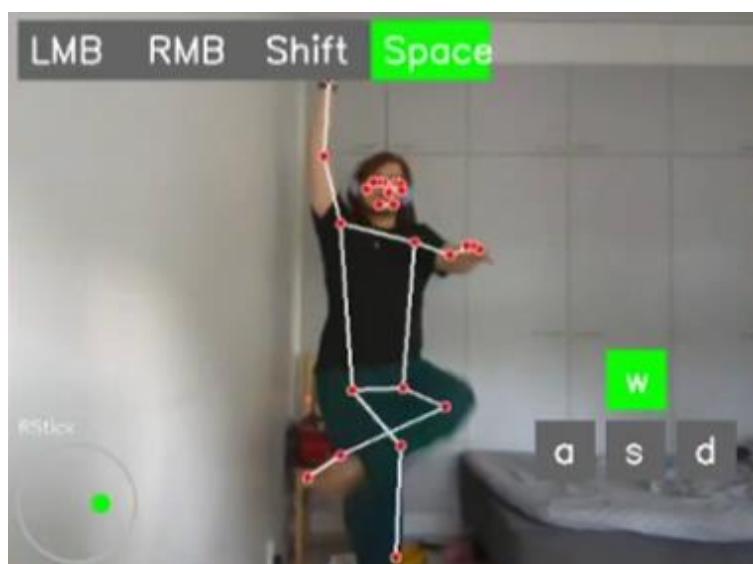
cap.release()
cv2.destroyAllWindows()

if __name__ == '__main__':
    main()

```

**Посилання на GitHub:** <https://github.com/SashaSuchkov/CamBoard>

**Тестування:** Програма була успішно протестована у відеогрі “ULTRAKILL”, де, при ввімкненій допомозі у прицілюванні у налаштуваннях гри, було при керуванні програмою пройдено рівень 0-4 на складності “Lenient”. Через динамічність гри можу зазначити, що гра таким чином була досить фізично активною. Оскільки гра створена на рушії Unity, можна допустити, що більшість ігор на цьому рушії з видом від першого обличчя та з підтримкою геймпаду можуть працювати з управлінням камерою.



*Rис. 1 – Натискання Space при затиснутому «Вперед» на лівому стіку у більш ранній версії програми із оверлеєм, де замість лівого стіку зображені клавіші WASD.*

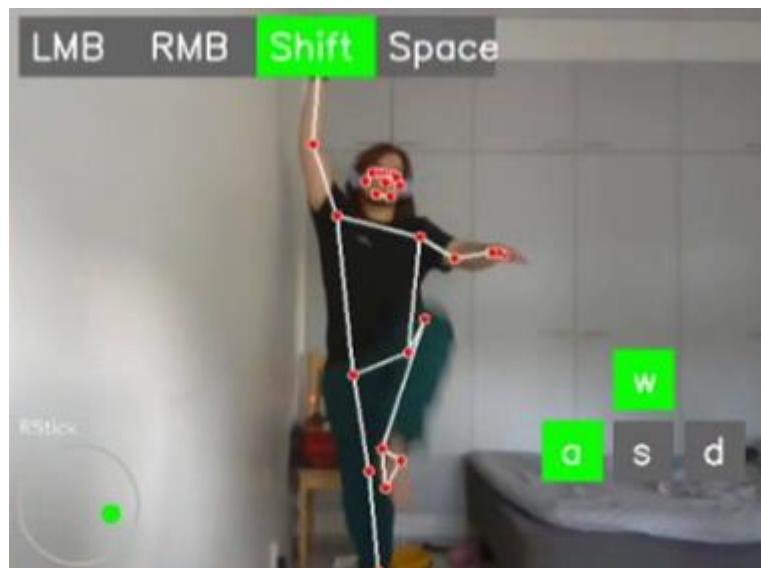


Рис. 2 – Натискання Shift у аналогічній до Рис. 1 ситуації.

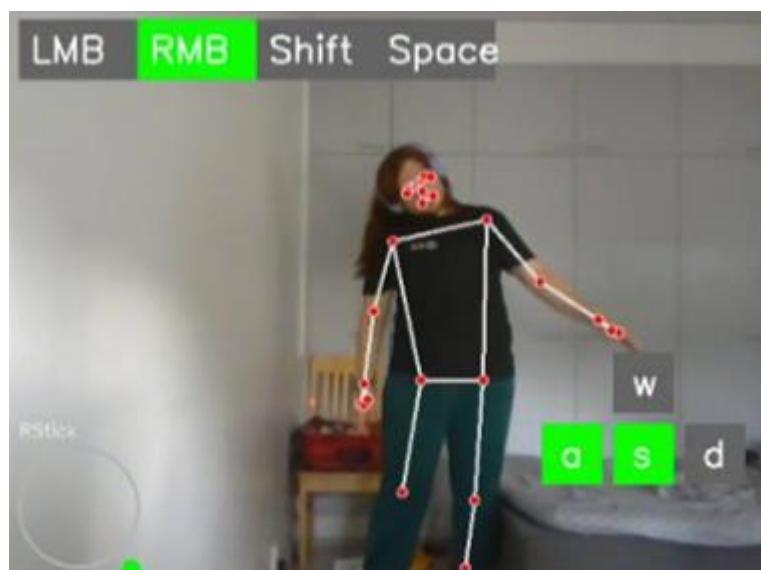


Рис. 3 – Натискання ПКМ при руху правого стіку вправо-вниз та лівого стіку вліво-вниз у версії програми з Рис. 1 і 2

**Висновок:** У результаті виконання курсової роботи було створено інтерактивну програму для управління відеограли за допомогою жестів тіла, що читаються через камеру. Програма успішно емулює лівий і правий стіки геймпада, кліки миші, клавіші Shift і Space. Рухи користувача відображаються у вигляді оверлею поверх відеопотоку. Система протестована на реальних іграх та демонструє високу стабільність і точність розпізнавання рухів, але може у майбутньому бути значно покращена та модифікована через open-source код, як мною, так і іншими людьми.