```
import cv2
import pygame
import mediapipe as mp
import numpy as np
from collections import deque
import math

# Initialize Pygame and window
pygame.init()
WIDTH, HEIGHT = 1280, 720
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Interactive Projection System")
clock = pygame.time.Clock()

# Sound
try:
    click_sound = pygame.mixer.Sound("click.wav")
except:
    click_sound = None

# Define grid buttons
buttons = []
cols, rows = 3, 2
padding = 40
bw, bh = 250, 100
color_list = [(255, 100, 100), (100, 255, 100), (100, 100, 255),
              (255, 255, 100), (255, 100, 255), (100, 255, 255)]

for row in range(rows):
    for col in range(cols):
        x = padding + col * (bw + padding)
        y = padding + row * (bh + padding)
        rect = pygame.Rect(x, y, bw, bh)
        buttons.append({
            "rect": rect,
            "default_color": color_list[row * cols + col],
            "hover_color": (200, 200, 200),
            "active_color": (255, 255, 255),
            "current_color": color_list[row * cols + col],
            "touched": False
        })

# Hand detection
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(max_num_hands=1,
                       min_detection_confidence=0.7,
                       min_tracking_confidence=0.7)
cap = cv2.VideoCapture(0)  # Change to 1 or 2 for external camera
smooth_buffer = deque(maxlen=5)

prev_x = None
gesture_text = ""
```

```
def distance(p1, p2):
    return math.hypot(p1[0] - p2[0], p1[1] - p2[1])

def detect_swipe(curr_x, prev_x):
    if prev_x is not None and abs(curr_x - prev_x) > 100:
        return "Swipe Left" if curr_x < prev_x else "Swipe Right"
    return ""

# Main loop
running = True
while running:
    screen.fill((30, 30, 30))
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(rgb)

    finger_pos = None
    gesture_text = ""

    if results.multi_hand_landmarks:
        hand_landmarks = results.multi_hand_landmarks[0]
        h, w, _ = frame.shape

        index_tip = hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP]
        thumb_tip = hand_landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP]

        x = int(index_tip.x * w)
        y = int(index_tip.y * h)
        smooth_buffer.append((x, y))

        avg_x = int(np.mean([p[0] for p in smooth_buffer]))
        avg_y = int(np.mean([p[1] for p in smooth_buffer]))

        # Gesture: open vs closed hand
        finger_dist = distance((index_tip.x, index_tip.y), (thumb_tip.x, thumb_tip.y))
        gesture_text = "Closed Hand" if finger_dist < 0.04 else "Open Hand"

        # Gesture: swipe detection
        swipe_gesture = detect_swipe(avg_x, prev_x)
        if swipe_gesture:
            gesture_text = swipe_gesture
        prev_x = avg_x

        # Map to pygame screen
        scale_x = WIDTH / w
        scale_y = HEIGHT / h
        px = int(avg_x * scale_x)
        py = int(avg_y * scale_y)
        finger_pos = (px, py)

        pygame.draw.circle(screen, (0, 255, 0), finger_pos, 12)
```

```
    # Draw and update buttons
    for btn in buttons:
        rect = btn["rect"]
        if finger_pos and rect.collidepoint(finger_pos):
            if not btn["touched"]:
                btn["touched"] = True
                btn["current_color"] = btn["active_color"]
                if click_sound:
                    click_sound.play()
            else:
                btn["current_color"] = btn["active_color"]
        else:
            btn["current_color"] = btn["default_color"]
            btn["touched"] = False

        pygame.draw.rect(screen, btn["current_color"], rect, border_radius=10)
        pygame.draw.rect(screen, (255, 255, 255), rect, width=2, border_radius=10)

    # Gesture text
    if gesture_text:
        font = pygame.font.SysFont("arial", 30)
        text = font.render(f"Gesture: {gesture_text}", True, (255, 255, 255))
        screen.blit(text, (WIDTH - 350, HEIGHT - 40))

    # Event handling (add Q key)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_q:
                running = False

    pygame.display.flip()
    clock.tick(30)

cap.release()
pygame.quit()
```