Event Handling System for Touchscreen Input

Objective: Simulate an event-driven system for processing touchscreen inputs on an HMI.

Requirements:

Create an Event class to represent user interactions:

Attributes: eventType (e.g., Tap, Swipe), x and y coordinates, and timestamp.

Implement an event queue using std::queue:

Store multiple events.

Process events one by one.

Handle specific events:

Tap: Display a message showing the tapped position.

Swipe: Calculate the swipe direction (up/down/left/right) and display it.

Simulate event generation:

Populate the queue with random events.

Deliverables:

A program simulating event processing.

Example output demonstrating event handling.

Program:

#include <iostream>

#include <queue>

#include <cstdlib>

#include <ctime>

#include <string>

#include <cmath>

using namespace std;

enum EventType

{

Tap,

Swipe

};

class Event

{

public:

EventType eventType;

int x, y;

time\_t timestamp;

Event(EventType type, int xPos, int yPos) : eventType(type), x(xPos), y(yPos), timestamp(time(0)) {}

void display() const

{

cout << "Event Type: " << (eventType == Tap ? "Tap" : "Swipe")

<< " | Coordinates: (" << x << ", " << y << ")"

<< " | Timestamp: " << timestamp << "\n";

}

};

class EventHandler

{

private:

queue<Event> eventQueue;

public:

void addEvent(const Event& event)

{

eventQueue.push(event);

}

void processEvents()

{

while (!eventQueue.empty())

{

Event currentEvent = eventQueue.front();

eventQueue.pop();

currentEvent.display();

if (currentEvent.eventType == Tap)

{

handleTap(currentEvent);

} else if (currentEvent.eventType == Swipe)

{

handleSwipe(currentEvent);

}

}

}

void handleTap(const Event& event)

{

cout << "Tapped at position (" << event.x << ", " << event.y << ")\n";

}

void handleSwipe(const Event& event)

{

int deltaX = event.x - 50;

int deltaY = event.y - 50;

if (abs(deltaX) > abs(deltaY))

{

if (deltaX > 0)

{

cout << "Swipe Right\n";

}

else

{

cout << "Swipe Left\n";

}

}

else

{

if (deltaY > 0)

{

cout << "Swipe Down\n";

}

else

{

cout << "Swipe Up\n";

}

}

}

};

void generateRandomEvent(EventHandler& handler)

{

EventType eventType = (rand() % 2 == 0) ? Tap : Swipe;

int x = rand() % 100;

int y = rand() % 100;

Event newEvent(eventType, x, y);

handler.addEvent(newEvent);

}

int main()

{

srand(static\_cast<unsigned int>(time(0)));

EventHandler handler;

for (int i = 0; i < 10; ++i)

{

generateRandomEvent(handler);

}

cout << "Processing events...\n";

handler.processEvents();

return 0;

}

Output:

Processing events...

Event Type: Swipe | Coordinates: (56, 30) | Timestamp: 1732604222

Swipe Up

Event Type: Swipe | Coordinates: (96, 57) | Timestamp: 1732604222

Swipe Right

Event Type: Swipe | Coordinates: (98, 69) | Timestamp: 1732604222

Swipe Right

Event Type: Swipe | Coordinates: (90, 39) | Timestamp: 1732604222

Swipe Right

Event Type: Tap | Coordinates: (5, 46) | Timestamp: 1732604222

Tapped at position (5, 46)

Event Type: Swipe | Coordinates: (41, 14) | Timestamp: 1732604222

Swipe Up

Event Type: Swipe | Coordinates: (28, 21) | Timestamp: 1732604222

Swipe Up

Event Type: Tap | Coordinates: (67, 54) | Timestamp: 1732604222

Tapped at position (67, 54)

Event Type: Swipe | Coordinates: (39, 25) | Timestamp: 1732604222

Swipe Up

Event Type: Swipe | Coordinates: (45, 16) | Timestamp: 1732604222

Swipe Up