documentation\digitalDevLogs\InputTracker.cpp

```cpp
//imports
#include <vector>
#include <fstream>
#include <string>
#include "../src/Globals.hpp"
#include "../src/Math/Vector2.hpp"

//using definitions
using std::vector;
using std::ifstream;
using std::getline;
using std::stoi;
using std::string;

//file globals
bool shouldTrack = false;
bool previousShouldTrack = shouldTrack;
vector<string> emulatedInputLines;



//file loading(INSIDE VEX INITIALIZE)
//load file and split input lines
string line;
ifstream inputFile(autonomousCodeLocation);
if (inputFile.is_open()){
    while (getline(inputFile, line))
    {
        emulatedInputLines.push_back(line);
    }
    inputFile.close();
};



//file parsing and executing (INSIDE VEX AUTONOMOUS)
//iterate through each input line
for (int i = 0; i < emulatedInputLines.size(); i++){
    //get input line string and process into emulated inputs
    string inputLine = emulatedInputLines.at(1);

    //get current input stage's emulated inputs
    int emulatedInput[16];
    int count = 0;
    string splitPart;
    while (getline(inputLine, splitPart, ",")){
        //convert line to int and store
        emulatedInput[count] = stoi(splitPart);
        //increment
        count++;
    }

    //call movement with emulated movement
    Movement(emulatedInput);

    //delay
    pros::delay(time_delay);
};



//tracking code and saving (INSIDE VEX OPERATOR CONTROL)
//setup input tracker
vector<InputState> inputStates;
int controllerInputs[16];
int trackerCount = 0;
while (true) {
    //get inputs
    controllerInputs = {
        controller.get_analog(ANALOG_LEFT_X),  //input #00
        controller.get_analog(ANALOG_LEFT_Y),  //input #01
        controller.get_analog(ANALOG_RIGHT_X), //input #02
        controller.get_analog(ANALOG_RIGHT_Y), //input #03
        controller.get_digital(DIGITAL_UP),    //input #04
        controller.get_digital(DIGITAL_DOWN),  //input #05
        controller.get_digital(DIGITAL_LEFT),  //input #06
        controller.get_digital(DIGITAL_RIGHT), //input #07
        controller.get_digital(DIGITAL_A),     //input #08
        controller.get_digital(DIGITAL_B),     //input #09
        controller.get_digital(DIGITAL_X),     //input #10
        controller.get_digital(DIGITAL_Y),     //input #11
        controller.get_digital(DIGITAL_L1),    //input #12
        controller.get_digital(DIGITAL_L2),    //input #13
        controller.get_digital(DIGITAL_R1),    //input #14
        controller.get_digital(DIGITAL_R2)     //input #15
    };

    //pass to movement function
    Movement(controllerInputs);

    //track input
    if (shouldTrack){
        //update previous should track
        previousShouldTrack = shouldTrack;

        //check if should end tracking
        if (trackerCount > 15 * 1000/time_delay) {
            shouldTrack = false;
        }
        trackerCount += 1;
        //update screen
        master.set_text("Auto: " + trackerCount);

        //track input
        InputState inputState = new InputState(master);
        inputStates.push_back(inputState);
    }else if((!shouldTrack) and previousShouldTrack){
        //just ended tracking should save our tracked inputs
        //update previous should track
        previousShouldTrack = false;

        //process tracked inputs into file output
        string trackedInputsOutput = "";
        for (int i = 0; i < inputStates.size(); i++){
            trackedInputsOutput.append(inputStates[i].CompileSaveLine() + "\n");
        }
        trackedInputsOutput.pop_back();

        //save processed tracked inputs
        FILE* usd_input_save = fopen(autonomousCodeLocation, "w");
        fputs(trackedInputsOutput, usd_input_save);
        fclose(usd_input_save);
    }

    //wait 20 milliseconds to next op control period
    pros::delay(time_delay);
};



//movement code(SEPARATE FILE)
void Movement(int[16] controllerInputs){
    //take in joystick inputs
    Vector2<int> leftJoystick = new Vector2<int>(controllerInputs[0], controllerInputs[1]);
    Vector2<int> rightJoystick = new Vector2<int>(controllerInputs[2], controllerInputs[3]);

    //update motors
    left_mtr.move(leftJoystick.getY());
    right_mtr.move(rightJoystick.getY());
}
```