```cpp
//includes
#include "main.h"
#include "display/lvgl.h"
#include "../src/Control/InputState.cpp"
#include "../src/Graphics/Button.cpp"
#include "../src/Math/Vector2.hpp"
#include <string>
#include <vector>
#include <fstream>

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

//definitions and globals
#define time_delay 20 //milliseconds
#define left_mtr_port 1
#define right_mtr_port 2
Button autonomousMenuButton;
Button mainMenuButton;
Button statusMenuButton;

pros::Controller master(pros::E_CONTROLLER_MASTER);
pros::Motor left_mtr(left_mtr_port);
pros::Motor right_mtr(right_mtr_port);

std::string autonomousCodeLocation = "/usd/autonomousMovement.routine";
bool shouldTrack = false;
bool previousShouldTrack = shouldTrack;
vector<string> emulatedInputLines;

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

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

//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();
};
```

```
57
58
59
60  //file parsing and executing (INSIDE VEX AUTONOMOUS)
61  //iterate through each input line
62  for (int i = 0; i < emulatedInputLines.size(); i++){
63      //get input line string and process into emulated inputs
64      string inputLine = emulatedInputLines.at(1);
65
66      //get current input stage's emulated inputs
67      int emulatedInput[16];
68      int count = 0;
69      string splitPart;
70      while (getline(inputLine, splitPart, ",")){
71          //convert line to int and store
72          emulatedInput[count] = stoi(splitPart);
73          //increment
74          count++;
75      }
76
77      //call movement with emulated movement
78      Movement(emulatedInput);
79
80      //delay
81      pros::delay(time_delay);
82  };
83
84
85
86  //tracking code and saving (INSIDE VEX OPERATOR CONTROL)
87  //iterate through each input line
88  for (int i = 0; i < emulatedInputLines.size(); i++){
89      //get input line string and process into emulated inputs
90      string inputLine = emulatedInputLines.at(i);
91
92      //get current input stage's emulated inputs
93      int emulatedInput[16];
94      int count = 0;
95      string separator = ",";
96      string splitPart;
97      size_t pos = 0;
98      while ((pos = inputLine.find(separator)) != std::string::npos) {
99          splitPart = inputLine.substr(0, pos);
100         std::cout << splitPart << std::endl;
101         inputLine.erase(0, pos + separator.length());
102     }
103
104     //call movement with emulated movement
105     Movement(emulatedInput);
106
107     //delay
108     pros::delay(time_delay);
109 }
110
111 //driver control mode
112 //setup input tracker
```

```
113 vector<InputState> inputStates;
114 int trackerCount = 0;
115 while (true) {
116     int controllerInputs[16] = {
117         master.get_analog(ANALOG_LEFT_X),  //input #00
118         master.get_analog(ANALOG_LEFT_Y),  //input #01
119         master.get_analog(ANALOG_RIGHT_X), //input #02
120         master.get_analog(ANALOG_RIGHT_Y), //input #03
121         master.get_digital(DIGITAL_UP),    //input #04
122         master.get_digital(DIGITAL_DOWN),  //input #05
123         master.get_digital(DIGITAL_LEFT),  //input #06
124         master.get_digital(DIGITAL_RIGHT), //input #07
125         master.get_digital(DIGITAL_A),     //input #08
126         master.get_digital(DIGITAL_B),     //input #09
127         master.get_digital(DIGITAL_X),     //input #10
128         master.get_digital(DIGITAL_Y),     //input #11
129         master.get_digital(DIGITAL_L1),    //input #12
130         master.get_digital(DIGITAL_L2),    //input #13
131         master.get_digital(DIGITAL_R1),    //input #14
132         master.get_digital(DIGITAL_R2)     //input #15
133     };
134
135     //pass to movement function
136     Movement(controllerInputs);
137
138     //track input
139     if (shouldTrack){
140         //update previous should track
141         previousShouldTrack = shouldTrack;
142
143         //check if should end tracking
144         if (trackerCount > 15 * 1000/time_delay) {
145             shouldTrack = false;
146         }
147         trackerCount += 1;
148
149         //update screen
150         //master.set_text("Auto: " + trackerCount);
151         //track input
152         InputState inputState;
153         inputStates.push_back(inputState);
154     }else if((!shouldTrack) and previousShouldTrack){
155         //just ended tracking should save our tracked inputs
156         //update previous should track
157         previousShouldTrack = false;
158
159         //process tracked inputs into file output
160         string trackedInputsOutput = "";
161         for (int i = 0; i < inputStates.size(); i++){
162             trackedInputsOutput.append(inputStates[i].CompileSaveLine() + "\n");
163         }
164         trackedInputsOutput.pop_back();
165
166         //save processed tracked inputs
167         FILE* usd_input_save = fopen(autonomousCodeLocation.c_str(), "w");
168         fputs(trackedInputsOutput.c_str(), usd_input_save);
```

```
169             fclose(usd_input_save);
170         }
171
172         //wait 20 milliseconds to next op control period
173         pros::delay(time_delay);
174     }
```