Nama : Gusti Gratia Delpiera

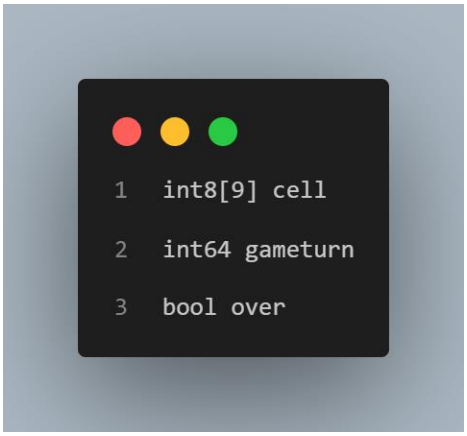
NRP : 5026231097

Laporan Final Project Magang Bayucaraka 2024

Link video demonstrasi :
https://drive.google.com/drive/folders/1QzTbVdO0mI9QPN13MNxOFOfdTONN8jro?usp=sharing

Langkah Pengerjaan :

1. Pada direktori FP, buat package yang diperlukan, disini saya membuat 3 package yaitu ‘vision’ untuk mengolah citra, ‘control’ untuk mengolah data dan algoritma, dan ‘griddata’ yang berfungsi sebagai custom message.
2. Package custom message memiliki message GameState yang berisi:
    a. Data dari Sembilan kolom (cell) : masing masing kolom memiliki nilai yaitu, 0 artinya kosong, 1 artinya telah diisi oleh player, 2 artinya telah diisi oleh bot.
    b. Gameturn : sebagai counter untuk kapan bot harus berjalan dan kapan player harus berjalan.
    c. Over : Sebagai penanda apakah game telah berakhir atau belum.



    Tambahkan dependencies yang diperlukan pada package.xml dan Cmakelist, setelah itu build custom message package.

3. Pada package vision, tambahkan dependencies yang dibutuhkan untuk pengolahan citra yaitu rclpy dan custom message (griddata), begitu juga pada package control.
4. Membuat Node pada package vision :
    a. Import library yang dibutuhkan, termasuk rclpy, GameState, dan opencv.

```python
import rclpy
from rclpy.node import Node
from griddata.msg import GameState
import cv2

```

b. Buat sebuah Node, di sini saya menggunakan MyNode
c. Dalam satu node itu, saya membuat publisher dan juga subscriber, yang mempublish topic **vision** dan subscribe ke topic **control**.

```python
# Periodic Timer
        self.timer_ = self.create_timer(0.1, self.publish_message)
        # Create Subscription and subscribe topic 'control'
        self.subscriber_ = self.create_subscription(GameState, 'control', self.timer_callback, 10)
        # Create Publisher and publish topic 'vision'
        self.publisher_ = self.create_publisher(GameState, 'vision', 10)

```

d. Buat data membernya, yang akan digunakan untuk menyimpan data sementara dalam Node.

```python
# Data members
self.cell = [0,0,0,0,0,0,0,0,0]
self.GameOver = 0
self.move = 5
self.Terminate = True
self.isTied = True
self.wincond = 0
self.turn = 0
```

e. Dengan opencv gunakan cv2.VideoCapture untuk membuka webcam, untuk indexnya sesuaikan dengan device yang digunakan, karena disini saya menggunakan device external maka, indexnya adalah 1.

```
# Open Webcam with external device
self.cap = cv2.VideoCapture(1)
```

f. Buat method untuk subscriptionnya dan ambil data dari message ke data member.

```
def timer_callback(self, msg):
    for i in range(0,9):
        self.cell[i] = msg.cell[i]
```

g. Buat method untuk publisher, dan send data dari data member ke GameState.msg.

```
def publish_message(self):
    # send data from data member into GameState.msg
    msg = GameState()
    for i in range(0,9):
        msg.cell[i] = self.cell[i]
    msg.gameturn = self.turn
    msg.over = self.Terminate
```

h. Ambil data dari webcam per framenya

```
# Read frame by frame
ret, frame = self.cap.read()

```

i. Buat kode untuk membuat sebuah lingkaran biru sebagai bot move per gridnya, karena disini saya menyimpan data grid pada sebuah array satu dimensi, maka tiap indeksnya merepresentasikan status dari satu kotak pada grid tersebut. Jika statusnya adalah 2, maka itu adalah botmove.

```
            # Draw Blue Circles as bot move
            if(msg.cell[0] == 2):
                cv2.circle(frame, (220,87), 55, (255,0,0), -1)
            if(msg.cell[1] == 2):
                cv2.circle(frame, (365,87), 55, (255,0,0), -1)
            if(msg.cell[2] == 2):
                cv2.circle(frame, (515,87), 55, (255,0,0), -1)
            if(msg.cell[3] == 2):
                cv2.circle(frame, (220,235), 55, (255,0,0), -1)
            if(msg.cell[4] == 2):
                cv2.circle(frame, (365,235), 55, (255,0,0), -1)
            if(msg.cell[5] == 2):
                cv2.circle(frame, (515,235), 55, (255,0,0), -1)
            if(msg.cell[6] == 2):
                cv2.circle(frame, (220,390), 55, (255,0,0), -1)
            if(msg.cell[7] == 2):
                cv2.circle(frame, (365,390), 55, (255,0,0), -1)
            if(msg.cell[8] == 2):
                cv2.circle(frame, (515,390), 55, (255,0,0), -1)

```

j. Buat kode program untuk mengecek apakah bot telah berakhir menang atau seri, jika menang maka isTied akan False dan Terminate akan false (fungsi terminate disini untuk menghentikan bot agar tidak melakukan move lagi). Wincond adalah kondisi kemenangan, dalam tictactoe, ada 8 kondisi : 3 vertikal, 3 horizontal, dan 2 diagonal.

```
# Check Win (continuous line of three in a row, column, or diagonal)
            if self.move!=0:
                # Row
                if(msg.cell[0] == msg.cell[1] and msg.cell[1] == msg.cell[2] and msg.cell[0] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 1
                elif(msg.cell[3] == msg.cell[4] and msg.cell[4] == msg.cell[5] and msg.cell[3] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 2
                elif(msg.cell[6] == msg.cell[7] and msg.cell[7] == msg.cell[8] and msg.cell[6] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 3
                # Column
                elif(msg.cell[0] == msg.cell[3] and msg.cell[3] == msg.cell[6] and msg.cell[0] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 4
                elif(msg.cell[1] == msg.cell[4] and msg.cell[4] == msg.cell[7] and msg.cell[1] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 5
                elif(msg.cell[2] == msg.cell[5] and msg.cell[5] == msg.cell[8] and msg.cell[2] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 6
                # Diagonal
                elif(msg.cell[0] == msg.cell[4] and msg.cell[4] == msg.cell[8] and msg.cell[0] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 7
                elif(msg.cell[2] == msg.cell[4] and msg.cell[4] == msg.cell[6] and msg.cell[2] == 2):
                    self.isTied = False
                    self.Terminate = False
                    self.wincond = 8
```

k. Jika bot menang maka inputkan text yang sesuai.

```
            # Game Over (Lose)
            if(self.isTied==False):
                if(self.wincond == 1):
                    cv2.line(frame, (215,95), (515, 95), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 2):
                    cv2.line(frame, (215,245), (515, 245), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 3):
                    cv2.line(frame, (215,395), (515, 395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 4):
                    cv2.line(frame, (215,95), (215, 395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 5):
                    cv2.line(frame, (365,95), (365, 395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 6):
                    cv2.line(frame, (515,95), (515, 395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 7):
                    cv2.line(frame, (215,95), (515,395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
                elif(self.wincond == 8):
                    cv2.line(frame, (515,95), (215,395), (0,255,255), 30)
                    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
                    cv2.putText(frame, "(YOU LOSE)", (50,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)

```

Jika seri, maka inputkan teks yang sesuai dan terminate akan False

```
# Game over (Tied)
elif (self.isTied == True and self.move == 0):
    cv2.putText(frame, "GAME OVER", (50,245), cv2.FONT_HERSHEY_TRIPLEX, 3, (7,17,20), 8)
    cv2.putText(frame, "(TIED)", (205,400), cv2.FONT_HERSHEY_SIMPLEX, 3, (36,45,48), 8)
    self.Terminate = False
```

l. Masukkan teks pada window “press ‘m’ to proceed” agar player bisa memahami apa yang harus dilakukan.

```
            # Add text
            cv2.putText(frame, "Press 'm'", (5,30), cv2.FONT_HERSHEY_DUPLEX, 1.2, (0,0,0), 1)
            cv2.putText(frame, "to proceed", (2,60), cv2.FONT_HERSHEY_DUPLEX, 1, (0,0,0), 1)

```

m. Jalankan cv2.imshow untuk menampilkan window

```
# Show the window
cv2.imshow("Autonomous Tic-Tac-Toe", frame)

```

n. Jalankan waitkey dan detections akan dieksekusi jika player menekan ‘m’ pada keyboard.

```
# Make the 'm' button as proceed button
if cv2.waitKey(1) & 0xFF == ord('m'):
    self.process_imagered(frame)
    self.process_imageblue(frame)
    self.turn += 2
    self.move -= 1
```

o. Buat method process_imagered, method ini akan melakukan detection pada warna merah, dengan convert frame menjadi hsv, lalu buat mask dengan lowerbound dan upperbound warna merah, deteksi contour dengan findContours, dan jika warna merah terdeteksi di dalam posisi kolom grid tertentu maka nilai dari kolom grid tersebut akan berubah menjadi 1. Begitu juga dengan method untuk detection warna biru, perbedaannya, kolom grid akan diubah menjadi 2.

```python
    # Detect the player mark (Red)
    def process_imagered(self, image):
        # Convert to hsv
        hsvred = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        # Create a Mask using defined color range in the frame
        maskred = cv2.inRange(hsvred, (0,80,120), (200,242,200))
        # Find Contours
        redcontours, hierarchy = cv2.findContours(maskred, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        if len(redcontours) != 0:
            for i in redcontours:
                # Find contour position and mark the grid position as 1 (red/Player mark)
                if cv2.contourArea(i)>500:
                    x1,y1,w,h = cv2.boundingRect(i)
                    # Left Column
                    if(x1 > 140 and (x1+w) < 290):
                        if(y1 > 20 and (y1+h) < 170):
                            self.cell[0] = 1
                        if(y1 > 170 and (y1+h) < 320):
                            self.cell[3] = 1
                        if(y1 > 320 and (y1+h) < 470):
                            self.cell[6] = 1
                    # Center Column
                    if(x1 > 290 and (x1+w) < 440):
                        if(y1 > 20 and (y1+h) < 170):
                            self.cell[1] = 1
                        if(y1 > 170 and (y1+h) < 320):
                            self.cell[4] = 1
                        if(y1 > 320 and (y1+h) < 470):
                            self.cell[7] = 1
                    # Right Column
                    if(x1 > 440 and (x1+w) < 590):
                        if(y1 > 20 and (y1+h) < 170):
                            self.cell[2] = 1
                        if(y1 > 170 and (y1+h) < 320):
                            self.cell[5] = 1
                        if(y1 > 320 and (y1+h) < 470):
                            self.cell[8] = 1
        self.get_logger().info("Processing Red . . .")

```

p. Spin node yang telah dibuat denganr rclpy.spin pada main function

```python
# Spin the node
def main(args=None):
    rclpy.init(args=args)
    publisher = MyNode()
    rclpy.spin(publisher)
    publisher.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

5. Membuat Node pada package control :
    a. Include semua library yang diperlukan

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/header.hpp"
#include <chrono>
#include "griddata/msg/game_state.hpp"
```

b. Di sini saya membuat node dengan nama MyNode, dan sama dengan Node di package vision, di sini saya membuat publisher dan subscriber dengan 2 topic berbeda. Subscribe ke topic **vision** untuk mendapatkan data dari webcam, dan publish data yang telah diolah di package ini ke topic **control.**

```
// Create Node
class MyNode : public rclcpp::Node {
    public:
        MyNode() : Node("pubcontrol"){
            // Subscribe vision topic
            subscription_ = this->create_subscription<griddata::msg::GameState>("vision",10,
            std::bind(&MyNode::timer_callback, this, std::placeholders::_1));
            // Publish control topic
            publisher_ = this->create_publisher<griddata::msg::GameState>("control", 10);
            //periodic timer
            timer_ = this->create_wall_timer(10000ms, std::bind(&MyNode::publish_msg, this));

        }
```

c. Buat data member untuk menyimpan data sementara pada node.

```
private:
    // Data members
    int cell[9];
    int turn;
```

d. Buat method untuk cek apakah game telah berakhir dengan kemenangan atau seri.

```
// Method to check the game status
bool GameOver(int board[]) {
    // Check for a win
    if ((board[0] != 0 && board[0] == board[1] && board[1] == board[2]) ||
        (board[3] != 0 && board[3] == board[4] && board[4] == board[5]) ||
        (board[6] != 0 && board[6] == board[7] && board[7] == board[8]) ||
        (board[0] != 0 && board[0] == board[3] && board[3] == board[6]) ||
        (board[1] != 0 && board[1] == board[4] && board[4] == board[7]) ||
        (board[2] != 0 && board[2] == board[5] && board[5] == board[8]) ||
        (board[0] != 0 && board[0] == board[4] && board[4] == board[8]) ||
        (board[2] != 0 && board[2] == board[4] && board[4] == board[6])) {
        return true; // Someone won
    }

    // Check for a tie
    for (int i = 0; i < 9; ++i) {
        if (board[i] == 0) return false; // Found an empty cell, game is not over yet
    }
    return true; // Board is full, it's a tie
}
```

e. Buat method evaluate, yang mereturn skor sementara player vs bot. Method ini digunakan sebagai base case dari recursive method berikutnya.

```
// Evaluate the game
int evaluate(int arr[9]){
    const int win_pattern[8][3] = {
        {0,1,2}, {3,4,5}, {6,7,8},
        {0,3,6}, {1,4,7}, {2,5,8},
        {0,4,8},{2,4,6}
    };

    for(int i = 0;i<8;i++){
        int countplayer = 0;
        int countbot = 0;
        for(int j = 0;j<3;j++){
            if(arr[win_pattern[i][j]]==1){
                countplayer++;
            }
            else if(arr[win_pattern[i][j]]==2){
                countbot++;
            }
        }
        if(countplayer == 3){
            return -10;
        }
        if(countbot == 3){
            return 10;
        }
    }
    return 0;
}
```

f. Buat method Minimax, yang menjadi algoritma kunci dari game tictactoe.

```cpp
// Minimax algorithm for tic-tac-toe
int minimax(int data[], int depth, bool isMaximize){
    if (GameOver(data) || depth == 5){
        return evaluate(data);
    }
    else {
        // maximizing player
        if(isMaximize){
            int maxEval = std::numeric_limits<int>::min();
            for(int i = 0;i<9;++i){
                if(data[i] == 0){
                    data[i] = 2;
                    // Recursive
                    int eval = minimax(data, depth+1, false);
                    data[i] = 0;
                    maxEval = std::max(maxEval, eval);
                }
            }
            return maxEval;
        }
        // Minimizing bot
        else{
            int minEval = std::numeric_limits<int>::max();
            for(int i = 0;i<9;++i){
                if(data[i] == 0){
                    data[i] = 1;
                    // Recursive
                    int eval = minimax(data, depth+1, true);
                    data[i] = 0;
                    minEval = std::min(minEval, eval);
                }
            }
            return minEval;
        }
    }

}
```

Algoritma minimax adalah metode pencarian pemilihan langkah optimal secara berbasis tree yang digunakan untuk memprediksi langkah terbaik yang dapat diambil oleh pemain pada giliran tertentu dalam permainan. Algoritma ini melakukan pencarian secara rekursif melalui semua kemungkinan langkah yang mungkin dilakukan oleh kedua pemain, dengan mengasumsikan bahwa lawan berusaha untuk memaksimalkan keuntungan mereka sendiri dan mencoba meminimalkan keuntungan yang mungkin diperoleh oleh pemain yang menggunakan algoritma ini. Dengan melakukan evaluasi pada keuntungan yang mungkin dihasilkan oleh setiap langkah yang mungkin, algoritma minimax memilih langkah terbaik yang meminimalkan risiko kekalahan dan maksimalkan peluang kemenangan.

g. Buat method subscription dan simpan data dari custom message ke data member.

```
// Subscription callback
void timer_callback(const griddata::msg::GameState::SharedPtr msg){
    // seubscribe message and store it into the data member
    for(int i =0 ;i<9;++i){
        cell[i] = msg->cell[i];
    }
```

h. Inisiasi untuk algoritma recursive minimax

```
// Initiate Minimax Algorithm
int bestmove = -1;
int besteval = std::numeric_limits<int>::min();
for(int i = 0;i<9;++i){
    if(cell[i] == 0){
        cell[i] = 2;
        int eval = minimax(cell, 0, false);
        cell[i] = 0;
        if(eval>besteval){
            besteval = eval;
            bestmove = i;
        }
    }
}
```

i. Jika msg.over bernilai true, maka eksekusi programnya, jika false maka game telah berhenti (bot berhenti melakukan move)

```
// if the game is not over yet, continue to proceed
if(msg->over){
    if(msg->gameturn % 2 == 0){
        cell[bestmove] = 2;
    }
}
```

j. Buat method publisher untuk publish message.

```
// Publishing
void publish_msg(){
    auto msg = griddata::msg::GameState();
    for(int i = 0;i<9;i++){
        msg.cell[i] = cell[i];
    }
    publisher_->publish(msg);

}
```

k. Spin Node yang telah dibuat

```
// Spin Node
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<MyNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

6. Setelah kedua node telah selesai, build kedua package, dan source, lalu run kedua package bersamaan.

Kendala dalam pengerjaan :

Dari segi input device (webcam) masih belum HD maka warna yang ditampilkan mungkin juga sedikit berbeda, dipengaruhi dari segi pencahayaan juga, sehingga color detection mungkin belum optimal pada warna merah yang asli. Masalah tersebut saya akali dengan menggunakan lowerbound dan upperbound yang saya buat sendiri dengan menggunakan hsv trackbar yang saya siapkan di program lain, sehingga program ini masih bisa berjalan dengan lancar.