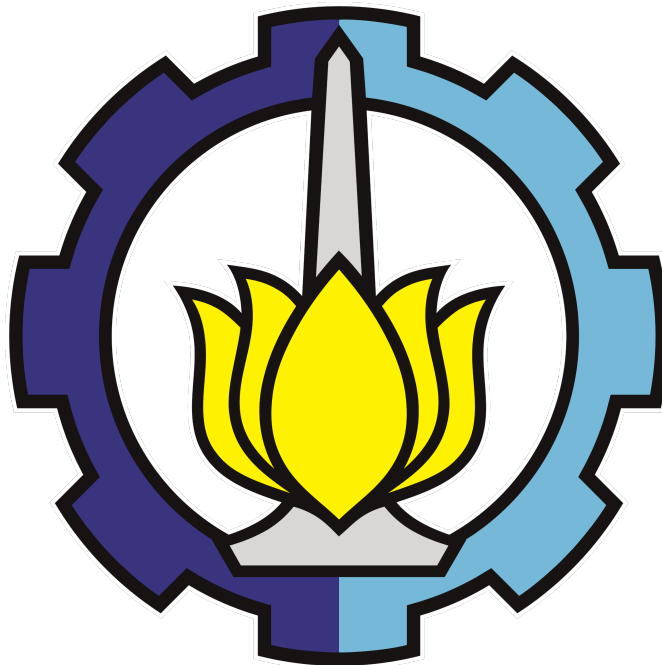


Laporan Tugas 3

Struktur Data dan Analisa Algoritma

Mochammad Hilmi Rusydiansyah
5024211008

October 5, 2022



Source Code Linked List Application with Stack

```
1 #include <iostream>
2 #include <unistd.h>
3
4 struct iniNode{
5     float data;
6     iniNode* below;
7 };
8
9 class Stack{
10 private:
11     iniNode* downNode;
12     iniNode* topNode;
13     int len;
14 public:
15     Stack(){
16         downNode = new iniNode;
17         topNode = new iniNode;
18         downNode->below = NULL;
19         topNode->below = NULL;
20         len = 0;
21     }
22
23     void push(float inp){
24         iniNode* temp = new iniNode;
25         if(!len){ // input saat stack kosong
26             temp->data = inp;
27             temp->below = NULL;
28
29             downNode = temp;
30             topNode = temp;
31         } else{ // input saat stack ada isinya
32             temp = topNode;
33
34             topNode = new iniNode;
35             topNode->data = inp;
36             topNode->below = temp;
37         }
38         len++;
39     }
40
41     void display(){
42         iniNode* temp = topNode;
43         std::cout << "ini top ygy" << std::endl;
44         while(temp){
45             std::cout << temp << " ==> " << temp->data << " -- " << temp
->below << std::endl;
46             temp = temp->below;
47         }
48         std::cout << "ini down ygy" << std::endl;
49     }
50 }
```

```

51     void pop(){
52         topNode = topNode->below;
53         len--;
54     }
55
56     bool isEmpty(){
57         if(!len){
58             return 1;
59         } return 0;
60     }
61
62     int size(){
63         return len;
64     }
65
66     iniNode* top(){
67         return topNode;
68     }
69 };
70
71 int main(){
72     Stack stackKuhh;
73     unsigned int sec_to_microsec = 1000000;
74
75     std::cout << "making a stack, please wait..." << std::endl;
76     usleep(2*sec_to_microsec);
77     std::cout << "completed !!!" << std::endl;
78     usleep(1*sec_to_microsec);
79
80     // isEmpty() checking
81     std::cout << std::endl << "check dengan isEmpty() : ";
82     if(stackKuhh.isEmpty()){
83         std::cout << stackKuhh.isEmpty() << " ---> ini stack khosong" <<
std::endl << std::endl;
84     }
85
86     std::cout << "Add 5 elements" << std::endl;
87     usleep(1*sec_to_microsec);
88     std::cout << "please wait..." << std::endl;
89     usleep(2*sec_to_microsec);
90     stackKuhh.push(5.0);
91     stackKuhh.push(6.0);
92     stackKuhh.push(7.0);
93     stackKuhh.push(8.0);
94     stackKuhh.push(9.0);
95     std::cout << "adding process completed" << std::endl << std::endl;
96     usleep(1*sec_to_microsec);
97
98     // displaying linked list
99     std::cout << "size() : " << stackKuhh.size() << std::endl;
100     stackKuhh.display();
101     usleep(2*sec_to_microsec);
102

```

```

103 // isEmpty() checking
104 std::cout << std::endl << "check dengan isEmpty() : ";
105 if(!stackKuhh.isEmpty()){
106     std::cout << stackKuhh.isEmpty() << " ---> sudah ada isinya" <<
std::endl;
107 }
108
109 // remove
110 std::cout << "\nPOP OUT 1 element" << std::endl;
111 usleep(1*sec_to_microsec);
112 stackKuhh.pop();
113 std::cout << "delete process completed" << std::endl;
114 usleep(1*sec_to_microsec);
115 std::cout << std::endl;
116
117 // displaying linked list
118 std::cout << "size() : " << stackKuhh.size() << std::endl;
119 stackKuhh.display();
120 usleep(2*sec_to_microsec);
121
122 std::cout << std::endl << "Add 7 elements" << std::endl;
123 usleep(1*sec_to_microsec);
124 std::cout << "please wait..." << std::endl;
125 usleep(2*sec_to_microsec);
126 stackKuhh.push(21);
127 stackKuhh.push(22);
128 stackKuhh.push(23);
129 stackKuhh.push(24);
130 stackKuhh.push(25);
131 stackKuhh.push(26);
132 stackKuhh.push(27);
133 std::cout << "adding process completed" << std::endl << std::endl;
134 usleep(1*sec_to_microsec);
135
136 // displaying linked list
137 std::cout << "size() : " << stackKuhh.size() << std::endl;
138 stackKuhh.display();
139 usleep(2*sec_to_microsec);
140
141 // remove
142 std::cout << "\nPOP OUT 4 element" << std::endl;
143 usleep(1*sec_to_microsec);
144 stackKuhh.pop();
145 stackKuhh.pop();
146 stackKuhh.pop();
147 stackKuhh.pop();
148 std::cout << "delete process completed" << std::endl;
149 usleep(1*sec_to_microsec);
150 std::cout << std::endl;
151
152 // displaying linked list
153 std::cout << "size() : " << stackKuhh.size() << std::endl;
154 stackKuhh.display();

```

```

155     usleep(2*sec_to_microsec);
156
157
158     // top
159     std::cout << std::endl << "check the element on the top" << std::endl
;
160     usleep(1*sec_to_microsec);
161     iniNode* nodeBaru = stackKuhh.top();
162     std::cout << "top -> data : " << nodeBaru->data << std::endl;
163     std::cout << "top -> address : " << nodeBaru << std::endl;
164
165     std::cout << "\nthank you" << std::endl;
166     usleep(1*sec_to_microsec);
167
168     return 0;
169 }

```

Source Code for Game Pesawat with SFML

```
1 #include <SFML/Graphics.hpp>
2 #include <cmath>
3 #include <unistd.h>
4
5 #define PI 3.14159265359
6
7 // global var////////////////////////////////////
8 unsigned int width_screen = 1000;
9 unsigned int height_screen = 1000;
10 //////////////////////////////////////
11
12 class PesawatLakon{
13 private :
14     float pos_x;
15     float pos_y;
16     float scale;
17     float body_points[13][2] = {{1,-3}, {0.25,-3}, {0.25,-4.5}, {0,-5},
18     {-0.25,-4.5}, {-0.25,-3}, {-1,-3}, {-1,1.5}, {-1.5,1.75}, {-1.25,2},
19     {1.25,2}, {1.5,1.75}, {1,1.5}}; // for drawing body_points plane
20     float wing_points[6][2] = {{1,-2}, {5,0}, {4,1}, {-4,1}, {-5,0},
21     {-1,-2}}; // for drawing wing_points plane
22     float tole_box;
23     float health;
24     float randValue_forBody[13];
25     float randValue_forWing[6];
26
27 public:
28     PesawatLakon(){
29         scale = 15;
30         pos_x = width_screen * 0.5;
31         pos_y = height_screen * 0.75;
32         tole_box = 2.5;
33         health = 100;
34
35         for(int i=0; i<13; i++){
36             randValue_forBody[i] = (rand()%3) * scale;
37         }
38         for(int i=0; i<6; i++){
39             randValue_forWing[i] = (rand()%3) * scale;
40         }
41
42     void moveHor(int speed){
43         pos_x = pos_x + speed;
44     }
45
46     void moveVer(int speed){
47         pos_y = pos_y + speed;
48     }
49
50     void drawPesawat(sf::RenderWindow& window){
```

```

49     sf::Vertex draw_body[] = {
50         sf::Vertex(sf::Vector2f(pos_x + body_points[0][0]*scale,
pos_y + body_points[0][1]*scale)),
51         sf::Vertex(sf::Vector2f(pos_x + body_points[1][0]*scale,
pos_y + body_points[1][1]*scale)),
52         sf::Vertex(sf::Vector2f(pos_x + body_points[2][0]*scale,
pos_y + body_points[2][1]*scale)),
53         sf::Vertex(sf::Vector2f(pos_x + body_points[3][0]*scale,
pos_y + body_points[3][1]*scale)),
54         sf::Vertex(sf::Vector2f(pos_x + body_points[4][0]*scale,
pos_y + body_points[4][1]*scale)),
55         sf::Vertex(sf::Vector2f(pos_x + body_points[5][0]*scale,
pos_y + body_points[5][1]*scale)),
56         sf::Vertex(sf::Vector2f(pos_x + body_points[6][0]*scale,
pos_y + body_points[6][1]*scale)),
57         sf::Vertex(sf::Vector2f(pos_x + body_points[7][0]*scale,
pos_y + body_points[7][1]*scale)),
58         sf::Vertex(sf::Vector2f(pos_x + body_points[8][0]*scale,
pos_y + body_points[8][1]*scale)),
59         sf::Vertex(sf::Vector2f(pos_x + body_points[9][0]*scale,
pos_y + body_points[9][1]*scale)),
60         sf::Vertex(sf::Vector2f(pos_x + body_points[10][0]*scale,
pos_y + body_points[10][1]*scale)),
61         sf::Vertex(sf::Vector2f(pos_x + body_points[11][0]*scale,
pos_y + body_points[11][1]*scale)),
62         sf::Vertex(sf::Vector2f(pos_x + body_points[12][0]*scale,
pos_y + body_points[12][1]*scale)),
63         sf::Vertex(sf::Vector2f(pos_x + body_points[0][0]*scale,
pos_y + body_points[0][1]*scale))
64     };
65     sf::Vertex draw_wing[] = {
66         sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
pos_y + wing_points[0][1]*scale)),
67         sf::Vertex(sf::Vector2f(pos_x + wing_points[1][0]*scale,
pos_y + wing_points[1][1]*scale)),
68         sf::Vertex(sf::Vector2f(pos_x + wing_points[2][0]*scale,
pos_y + wing_points[2][1]*scale)),
69         sf::Vertex(sf::Vector2f(pos_x + wing_points[3][0]*scale,
pos_y + wing_points[3][1]*scale)),
70         sf::Vertex(sf::Vector2f(pos_x + wing_points[4][0]*scale,
pos_y + wing_points[4][1]*scale)),
71         sf::Vertex(sf::Vector2f(pos_x + wing_points[5][0]*scale,
pos_y + wing_points[5][1]*scale)),
72         sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
pos_y + wing_points[0][1]*scale))
73     };
74
75     window.draw(draw_body, 13+1, sf::LineStrip);
76     window.draw(draw_wing, 6+1, sf::LineStrip);
77 }
78
79 void drawPesawatHancur(sf::RenderWindow& window){
80     // ini asalnya body

```

```

81         sf::Vertex draw_line1[] = {
82             sf::Vertex(sf::Vector2f((pos_x + body_points[0][0]*scale) +
                                     randValue_forBody[0],
                                     (pos_y + body_points[0][1]*
                                     scale) + randValue_forBody[0])),
83             sf::Vertex(sf::Vector2f((pos_x + body_points[1][0]*scale) +
                                     randValue_forBody[0],
                                     (pos_y + body_points[1][1]*
                                     scale) + randValue_forBody[0]))
84         };
85         sf::Vertex draw_line2[] = {
86             sf::Vertex(sf::Vector2f((pos_x + body_points[1][0]*scale) +
                                     randValue_forBody[1],
                                     (pos_y + body_points[1][1]*
                                     scale) + randValue_forBody[1])),
87             sf::Vertex(sf::Vector2f((pos_x + body_points[2][0]*scale) +
                                     randValue_forBody[1],
                                     (pos_y + body_points[2][1]*
                                     scale) + randValue_forBody[1]))
88         };
89         sf::Vertex draw_line3[] = {
90             sf::Vertex(sf::Vector2f((pos_x + body_points[2][0]*scale) -
                                     randValue_forBody[2],
                                     (pos_y + body_points[2][1]*
                                     scale) + randValue_forBody[2])),
91             sf::Vertex(sf::Vector2f((pos_x + body_points[3][0]*scale) -
                                     randValue_forBody[2],
                                     (pos_y + body_points[3][1]*
                                     scale) + randValue_forBody[2]))
92         };
93         sf::Vertex draw_line4[] = {
94             sf::Vertex(sf::Vector2f((pos_x + body_points[3][0]*scale) +
                                     randValue_forBody[3],
                                     (pos_y + body_points[3][1]*
                                     scale) - randValue_forBody[3])),
95             sf::Vertex(sf::Vector2f((pos_x + body_points[4][0]*scale) +
                                     randValue_forBody[3],
                                     (pos_y + body_points[4][1]*
                                     scale) - randValue_forBody[3]))
96         };
97         sf::Vertex draw_line5[] = {
98             sf::Vertex(sf::Vector2f((pos_x + body_points[4][0]*scale) -
                                     randValue_forBody[4],
                                     (pos_y + body_points[4][1]*
                                     scale) - randValue_forBody[4])),
99             sf::Vertex(sf::Vector2f((pos_x + body_points[5][0]*scale) -
                                     randValue_forBody[4],
                                     (pos_y + body_points[5][1]*
                                     scale) - randValue_forBody[4]))
100         };
101         sf::Vertex draw_line6[] = {
102             sf::Vertex(sf::Vector2f((pos_x + body_points[5][0]*scale) +
                                     randValue_forBody[5],
                                     (pos_y + body_points[5][1]*
                                     scale) - randValue_forBody[5])),
103             sf::Vertex(sf::Vector2f((pos_x + body_points[6][0]*scale) +
                                     randValue_forBody[5],
                                     (pos_y + body_points[6][1]*
                                     scale) - randValue_forBody[5]))
104         };
105         sf::Vertex draw_line7[] = {
106             sf::Vertex(sf::Vector2f((pos_x + body_points[6][0]*scale) +
                                     randValue_forBody[6],
                                     (pos_y + body_points[6][1]*
                                     scale) + randValue_forBody[6])),
107             sf::Vertex(sf::Vector2f((pos_x + body_points[7][0]*scale) +

```



```

108     randValue_forBody[6], (pos_y + body_points[7][1]*
109     scale) + randValue_forBody[6]))
110 };
111 sf::Vertex draw_line8[] = {
112     sf::Vertex(sf::Vector2f((pos_x + body_points[7][0]*scale) -
113     randValue_forBody[7], (pos_y + body_points[7][1]*
114     scale) + randValue_forBody[7])),
115     sf::Vertex(sf::Vector2f((pos_x + body_points[8][0]*scale) -
116     randValue_forBody[7], (pos_y + body_points[8][1]*
117     scale) + randValue_forBody[7]))
118 };
119 sf::Vertex draw_line9[] = {
120     sf::Vertex(sf::Vector2f((pos_x + body_points[8][0]*scale) -
121     randValue_forBody[8], (pos_y + body_points[8][1]*
122     scale) - randValue_forBody[8])),
123     sf::Vertex(sf::Vector2f((pos_x + body_points[9][0]*scale) -
124     randValue_forBody[8], (pos_y + body_points[9][1]*
125     scale) - randValue_forBody[8]))
126 };
127 sf::Vertex draw_line10[] = {
128     sf::Vertex(sf::Vector2f((pos_x + body_points[9][0]*scale) +
129     randValue_forBody[9], (pos_y + body_points[9][1]*
130     scale) + randValue_forBody[9])),
131     sf::Vertex(sf::Vector2f((pos_x + body_points[10][0]*scale) +
132     randValue_forBody[9], (pos_y + body_points[10][1]*
133     scale) + randValue_forBody[9]))
134 };
135 sf::Vertex draw_line11[] = {
136     sf::Vertex(sf::Vector2f((pos_x + body_points[10][0]*scale) +
137     randValue_forBody[10], (pos_y + body_points
138     [10][1]*scale) - randValue_forBody[10])),
139     sf::Vertex(sf::Vector2f((pos_x + body_points[11][0]*scale) +
140     randValue_forBody[10], (pos_y + body_points
141     [11][1]*scale) - randValue_forBody[10]))
142 };
143 sf::Vertex draw_line12[] = {
144     sf::Vertex(sf::Vector2f((pos_x + body_points[11][0]*scale) -
145     randValue_forBody[11], (pos_y + body_points
146     [11][1]*scale) - randValue_forBody[11])),
147     sf::Vertex(sf::Vector2f((pos_x + body_points[12][0]*scale) -
148     randValue_forBody[11], (pos_y + body_points
149     [12][1]*scale) - randValue_forBody[11]))
150 };
151 sf::Vertex draw_line13[] = {
152     sf::Vertex(sf::Vector2f((pos_x + body_points[12][0]*scale) -
153     randValue_forBody[12], (pos_y + body_points
154     [12][1]*scale) + randValue_forBody[12])),
155     sf::Vertex(sf::Vector2f((pos_x + body_points[0][0]*scale) -
156     randValue_forBody[12], (pos_y + body_points
157     [0][1]*scale) + randValue_forBody[12]))
158 };
159 //ini asalnya wing
160 sf::Vertex draw_line14[] = {

```

```

135         sf::Vertex(sf::Vector2f((pos_x + wing_points[0][0]*scale) +
randValue_forWing[0],
                                (pos_y + wing_points[0][1]*
scale) + randValue_forWing[0])),
136         sf::Vertex(sf::Vector2f((pos_x + wing_points[1][0]*scale) +
randValue_forWing[0],
                                (pos_y + wing_points[1][1]*
scale) + randValue_forWing[0])),
137     };
138     sf::Vertex draw_line15[] = {
139         sf::Vertex(sf::Vector2f((pos_x + wing_points[1][0]*scale) +
randValue_forWing[1],
                                (pos_y + wing_points[1][1]*
scale) + randValue_forWing[1])),
140         sf::Vertex(sf::Vector2f((pos_x + wing_points[2][0]*scale) +
randValue_forWing[1],
                                (pos_y + wing_points[2][1]*
scale) + randValue_forWing[1])),
141     };
142     sf::Vertex draw_line16[] = {
143         sf::Vertex(sf::Vector2f((pos_x + wing_points[2][0]*scale) -
randValue_forWing[2],
                                (pos_y + wing_points[2][1]*
scale) + randValue_forWing[2])),
144         sf::Vertex(sf::Vector2f((pos_x + wing_points[3][0]*scale) -
randValue_forWing[2],
                                (pos_y + wing_points[3][1]*
scale) + randValue_forWing[2])),
145     };
146     sf::Vertex draw_line17[] = {
147         sf::Vertex(sf::Vector2f((pos_x + wing_points[3][0]*scale) +
randValue_forWing[3],
                                (pos_y + wing_points[3][1]*
scale) - randValue_forWing[3])),
148         sf::Vertex(sf::Vector2f((pos_x + wing_points[4][0]*scale) +
randValue_forWing[3],
                                (pos_y + wing_points[4][1]*
scale) - randValue_forWing[3])),
149     };
150     sf::Vertex draw_line18[] = {
151         sf::Vertex(sf::Vector2f((pos_x + wing_points[4][0]*scale) -
randValue_forWing[4],
                                (pos_y + wing_points[4][1]*
scale) - randValue_forWing[4])),
152         sf::Vertex(sf::Vector2f((pos_x + wing_points[5][0]*scale) -
randValue_forWing[4],
                                (pos_y + wing_points[5][1]*
scale) - randValue_forWing[4])),
153     };
154     sf::Vertex draw_line19[] = {
155         sf::Vertex(sf::Vector2f((pos_x + wing_points[5][0]*scale) +
randValue_forWing[5],
                                (pos_y + wing_points[5][1]*
scale) - randValue_forWing[5])),
156         sf::Vertex(sf::Vector2f((pos_x + wing_points[0][0]*scale) +
randValue_forWing[5],
                                (pos_y + wing_points[0][1]*
scale) - randValue_forWing[5])),
157     };
158     // sf::Vertex draw_wing[] = {
159         // sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
pos_y + wing_points[0][1]*scale)),
160         // sf::Vertex(sf::Vector2f(pos_x + wing_points[1][0]*scale,
pos_y + wing_points[1][1]*scale)),
161         // sf::Vertex(sf::Vector2f(pos_x + wing_points[2][0]*scale,

```

```

162     pos_y + wing_points[2][1]*scale)),
        // sf::Vertex(sf::Vector2f(pos_x + wing_points[3][0]*scale,
163     pos_y + wing_points[3][1]*scale)),
        // sf::Vertex(sf::Vector2f(pos_x + wing_points[4][0]*scale,
164     pos_y + wing_points[4][1]*scale)),
        // sf::Vertex(sf::Vector2f(pos_x + wing_points[5][0]*scale,
165     pos_y + wing_points[5][1]*scale)),
        // sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
166     pos_y + wing_points[0][1]*scale))
167     // };
168
169     window.draw(draw_line1, 2, sf::Lines);
170     window.draw(draw_line2, 2, sf::Lines);
171     window.draw(draw_line3, 2, sf::Lines);
172     window.draw(draw_line4, 2, sf::Lines);
173     window.draw(draw_line5, 2, sf::Lines);
174     window.draw(draw_line6, 2, sf::Lines);
175     window.draw(draw_line7, 2, sf::Lines);
176     window.draw(draw_line8, 2, sf::Lines);
177     window.draw(draw_line9, 2, sf::Lines);
178     window.draw(draw_line10, 2, sf::Lines);
179     window.draw(draw_line11, 2, sf::Lines);
180     window.draw(draw_line12, 2, sf::Lines);
181     window.draw(draw_line13, 2, sf::Lines);
182     window.draw(draw_line14, 2, sf::Lines);
183     window.draw(draw_line15, 2, sf::Lines);
184     window.draw(draw_line16, 2, sf::Lines);
185     window.draw(draw_line17, 2, sf::Lines);
186     window.draw(draw_line18, 2, sf::Lines);
187     window.draw(draw_line19, 2, sf::Lines);
188     // window.draw(draw_wing, 6+1, sf::LineStrip);
189
190     for(int i=0; i<13; i++){
191         if(i%2==0){
192             randValue_forBody[i] += 1*0.025*scale;
193         } else{
194             randValue_forBody[i] -= 1*0.025*scale;
195         }
196     }
197     for(int i=0; i<6; i++){
198         if(i%2==0){
199             randValue_forWing[i] += 1*0.025*scale;
200         } else{
201             randValue_forWing[i] -= 1*0.025*scale;
202         }
203     }
204
205     float batasBox_kiri(){
206         return pos_x - tole_box * scale;
207     }
208     float batasBox_kanan(){
209         return pos_x + tole_box * scale;

```

```

210     }
211     float batasBox_atas(){
212         return pos_y - tole_box * scale;
213     }
214     float batasBox_bawah(){
215         return pos_y + tole_box * scale;
216     }
217
218     void reduceHealth(float damage){
219         if(health <= 0){
220             health = 0;
221         } else{
222             health = health + damage;
223         }
224     }
225
226     void displayHealth(sf::RenderWindow& window){
227         sf::RectangleShape rectangle(sf::Vector2f((health/100) * 220.f,
30.f));
228         rectangle.setPosition(sf::Vector2f(30.f, 940.f));
229
230         window.draw(rectangle);
231     }
232
233     float getHealth(){
234         return health;
235     }
236 };
237
238 class PesawatMusuh{
239 private :
240     float pos_x;
241     float pos_y;
242     float scale;
243     // int speed_y; // determine plane speed y
244     int speed_x; // determine plane speed x
245     float body_points[9][2] = {{0.5,0}, {0.5,-1.25}, {0,-1.5},
{-0.5,-1.25}, {-0.5,0}, {-0.25,2.75}, {-1,3}, {1,3}, {0.25,2.75}}; //
for drawing body_points plane
246     float wing_points[6][2] = {{2.5,0.5}, {3,1.75}, {2.5,-0.5},
{-2.5,-0.5}, {-3,1.75}, {-2.5,0.5}}; // for drawing wing_points plane
247
248     bool bullet_exist;
249     float pos_bullet_x;
250     float pos_bullet_y;
251     float rad_bullet;
252     int speed_bullet_y;
253
254 public:
255     PesawatMusuh(){
256         scale = rand()%15 + 5;
257         // speed_y = rand()%5 + 1;
258         speed_x = rand()%5 + 1;

```

```

259     pos_x = rand() % width_screen + 10;
260     pos_y = rand() % int(height_screen*0.25) + 10;
261
262     bullet_exist = false;
263     rad_bullet = 5;
264     speed_bullet_y = 5;
265 }
266
267 void moveRight(){
268     pos_x = pos_x + speed_x;
269     if(pos_x > width_screen){
270         pos_x = pos_x - width_screen;
271     }
272 }
273
274 void moveLeft(){
275     pos_x = pos_x - speed_x;
276     if(pos_x < 0){
277         pos_x = pos_x + width_screen;
278     }
279 }
280
281 void rotatePesawat(float degree){
282     int num_bpoints = 9;
283     int num_wpoints = 6;
284     // convert degree to radian
285     float rad = degree * (PI/180);
286
287     // rotating body
288     for(int i=0; i<num_bpoints; i++){
289         float temp[2]; // for storing one-point coordinate
290         // assign to temp
291         for(int j=0; j<2; j++){
292             temp[j] = body_points[i][j];
293         }
294         // convert cartesian to polar (r, theta)
295         float r = sqrt( pow(temp[0],2) + pow(temp[1],2) );
296         float theta = atan2(temp[1], temp[0]);
297
298         // adding theta by degree(radian) inputted
299         theta = theta + rad;
300
301         // convert polar to cartesian again and store it to temp
302         temp[0] = r * cos(theta); // as x
303         temp[1] = r * sin(theta); // as y
304
305         // return temp to body_array
306         for(int j=0; j<2; j++){
307             body_points[i][j] = temp[j];
308         }
309     }
310
311     // rotating wing

```

```

312     for(int i=0; i<num_wpoints; i++){
313         float temp[2]; // for storing one-point coordinate
314         // assign to temp
315         for(int j=0; j<2; j++){
316             temp[j] = wing_points[i][j];
317         }
318         // convert cartesian to polar (r, theta)
319         float r = sqrt( pow(temp[0],2) + pow(temp[1],2) );
320         float theta = atan2(temp[1], temp[0]);
321
322         // adding theta by degree(radian) inputted
323         theta = theta + rad;
324
325         // convert polar to cartesian again and store it to temp
326         temp[0] = r * cos(theta); // as x
327         temp[1] = r * sin(theta); // as y
328
329         // return temp to wing_array
330         for(int j=0; j<2; j++){
331             wing_points[i][j] = temp[j];
332         }
333     }
334 }
335
336 void drawPesawat(sf::RenderWindow& window){
337     sf::Vertex draw_body[] = {
338         sf::Vertex(sf::Vector2f(pos_x + body_points[0][0]*scale,
339 pos_y + body_points[0][1]*scale)),
340         sf::Vertex(sf::Vector2f(pos_x + body_points[1][0]*scale,
341 pos_y + body_points[1][1]*scale)),
342         sf::Vertex(sf::Vector2f(pos_x + body_points[2][0]*scale,
343 pos_y + body_points[2][1]*scale)),
344         sf::Vertex(sf::Vector2f(pos_x + body_points[3][0]*scale,
345 pos_y + body_points[3][1]*scale)),
346         sf::Vertex(sf::Vector2f(pos_x + body_points[4][0]*scale,
347 pos_y + body_points[4][1]*scale)),
348         sf::Vertex(sf::Vector2f(pos_x + body_points[5][0]*scale,
349 pos_y + body_points[5][1]*scale)),
350         sf::Vertex(sf::Vector2f(pos_x + body_points[6][0]*scale,
351 pos_y + body_points[6][1]*scale)),
352         sf::Vertex(sf::Vector2f(pos_x + body_points[7][0]*scale,
353 pos_y + body_points[7][1]*scale)),
354         sf::Vertex(sf::Vector2f(pos_x + body_points[8][0]*scale,
355 pos_y + body_points[8][1]*scale)),
356         sf::Vertex(sf::Vector2f(pos_x + body_points[0][0]*scale,
357 pos_y + body_points[0][1]*scale))
358     };
359     sf::Vertex draw_wing[] = {
360         sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
361 pos_y + wing_points[0][1]*scale)),
362         sf::Vertex(sf::Vector2f(pos_x + wing_points[1][0]*scale,
363 pos_y + wing_points[1][1]*scale)),
364         sf::Vertex(sf::Vector2f(pos_x + wing_points[2][0]*scale,
365 pos_y + wing_points[2][1]*scale))
366     };

```

```

353     pos_y + wing_points[2][1]*scale)),
        sf::Vertex(sf::Vector2f(pos_x + wing_points[3][0]*scale,
354     pos_y + wing_points[3][1]*scale)),
        sf::Vertex(sf::Vector2f(pos_x + wing_points[4][0]*scale,
355     pos_y + wing_points[4][1]*scale)),
        sf::Vertex(sf::Vector2f(pos_x + wing_points[5][0]*scale,
356     pos_y + wing_points[5][1]*scale)),
        sf::Vertex(sf::Vector2f(pos_x + wing_points[0][0]*scale,
357     pos_y + wing_points[0][1]*scale))
        };
358
359     window.draw(draw_body, 9+1, sf::LineStrip);
360     window.draw(draw_wing, 6+1, sf::LineStrip);
361 }
362
363 void saveBulletPos(){
364     pos_bullet_x = pos_x;
365     pos_bullet_y = pos_y;
366 }
367
368 void moveBulletDown(){
369     pos_bullet_y = pos_bullet_y + speed_bullet_y;
370 }
371
372 void drawBullet(sf::RenderWindow& window){
373     sf::CircleShape circle(rad_bullet);
374     circle.setPosition(sf::Vector2f(pos_bullet_x, pos_bullet_y));
375     window.draw(circle);
376 }
377
378 float posBulletX(){
379     return pos_bullet_x;
380 }
381
382 float posBulletY(){
383     return pos_bullet_y;
384 }
385
386 bool isBulletExist(){
387     return bullet_exist;
388 }
389
390 void setBulletExistence(bool existence){
391     bullet_exist = existence;
392 }
393 };
394
395
396 int main(){
397     sf::RenderWindow window(sf::VideoMode(width_screen, height_screen), "
Pesawat-Pesawat");
398
399     // create pesawat lakon

```

```

400     PesawatLakon jatayu;
401
402     // create pesawat musuh
403     int num_opponents = 6;
404     PesawatMusuh opponents[num_opponents];
405     // PesawatMusuh opponents_toLeft[num_opponents];
406     for(int i=0; i<num_opponents; i++){
407         if(i<(num_opponents/2)){
408             opponents[i].rotatePesawat(90);
409         } else{
410             opponents[i].rotatePesawat(-90);
411         }
412     }
413
414     while(window.isOpen()){
415         sf::Event event;
416         while(window.pollEvent(event)){
417             if(event.type == sf::Event::Closed){
418                 window.close();
419             }
420             else if(event.type == sf::Event::KeyPressed){
421                 if(event.key.code == sf::Keyboard::W){
422                     // printf("W\n");
423                     jatayu.moveVer(-10);
424                 } if(event.key.code == sf::Keyboard::S){
425                     // printf("S\n");
426                     jatayu.moveVer(10);
427                 } if(event.key.code == sf::Keyboard::A){
428                     // printf("A\n");
429                     jatayu.moveHor(-10);
430                 } if(event.key.code == sf::Keyboard::D){
431                     // printf("D\n");
432                     jatayu.moveHor(10);
433                 }
434             }
435             else if(event.type == sf::Event::MouseButtonPressed){
436                 if(event.mouseButton.button == sf::Mouse::Left){
437                     // printf("klik kiri\n");
438                 }
439             }
440         }
441
442         // kalo peluru kena ke kita
443         for(int i=0; i<num_opponents; i++){
444             if(opponents[i].posBulletX() >= jatayu.batasBox_kiri() &&
445                opponents[i].posBulletX() <= jatayu.batasBox_kanan() &&
446                opponents[i].posBulletY() >= jatayu.batasBox_atas() &&
447                opponents[i].posBulletY() <= jatayu.batasBox_bawah()){
448                 opponents[i].setBulletExistance(false);
449                 jatayu.reduceHealth(-25);
450             }
451         }
452

```



```

453     window.clear();
454
455     jatayu.displayHealth(window);
456     if(jatayu.getHealth() == 0){
457         jatayu.drawPesawatHancur(window);
458     } else{
459         jatayu.drawPesawat(window);
460     }
461
462     for(int i=0; i<num_opponents; i++){
463         // for draw pesawat musuh
464         if(i<(num_opponents/2)){
465             opponents[i].moveRight();
466         } else{
467             opponents[i].moveLeft();
468         }
469         opponents[i].drawPesawat(window);
470
471         // for draw bullet musuh
472         if(!(opponents[i].isBulletExist())){ // checking bullet
existence
473             opponents[i].saveBulletPos();
474             opponents[i].setBulletExistance(true);
475         }
476         opponents[i].moveBulletDown();
477         if(opponents[i].posBulletY() > width_screen){
478             opponents[i].saveBulletPos(); // reset bullet position
479         }
480         opponents[i].drawBullet(window);
481     }
482
483     window.display();
484     usleep(20*1000);
485 }
486
487 return 0;
488 }

```