

```
1  //-----
2  //      Title:MASTER DEGREE THESIS by ANTONIO SCAZZI
3  //
4  //  Description:Addon for Simconnect linked to Prepar3D
5  //      It has to be attached to a DRONE
6  //-----
7
8  #include "headers/globalvar.h"
9  #include "headers/readwrite.h"
10 #include "headers/dispatchfun.h"
11 #include "headers/autopilots.h"
12 #include "headers/miscellaneous.h"
13
14 //title of the addon
15 const char* TITLE_STRING = "DRONE PLANE";
16
17 //main function of the addon
18 int __cdecl _tmain(int argc, _TCHAR* argv[])
19 {
20     //lettura file configs.txt
21     LetturaConfigs();
22
23     // Apertura del simconnect
24     if (SUCCEEDED(SimConnect_Open(&hSimConnect, TITLE_STRING, NULL, 0, 0, 0)))
25     {
26         //connesso al simulatore
27         printf("\nConnected to Prepar3D");
28
29
30         //ciclo principale dell'applicazione
31         while (0 == flag_quit)
32         {
33             //funzione che gestisce gli eventi del simulatore
```

```
34     SimConnect_CallDispatch(hSimConnect, MyDispatchProc, NULL);
35
36
37     //controllo se il simulatore è in pausa o meno
38     if (flag_isrunning == 1)
39     {
40         //request data on user
41         hr = SimConnect_RequestDataOnSimObjectType(hSimConnect, REQUEST_1, DEFINITION_1, 0, 7
            SIMCONNECT_SIMOBJECT_TYPE_USER);
42
43         //request data on other plane
44         hr = SimConnect_RequestDataOnSimObjectType(hSimConnect, REQUEST_0, DEFINITION_1, 100000, 7
            SIMCONNECT_SIMOBJECT_TYPE_AIRCRAFT);
45
46         //richiesta di comando
47         RecieveCommand();
48
49         //ceck if is on ground and correct the position
50         IsOnGround();
51
52         //controllo del carrello e retrazione
53         GearCheck();
54
55         // autopilota C2, manovra di decollo e immissione in crociera
56         if (flag_initialgroundcheck == 1)
57         {
58             switch (flag_decollo)
59             {
60                 //inizializzazione
61                 case 1:
62                 {
63                     // manetta al 90%
64                     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_THROTTLE_SET, 7
```

```
14743, SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
65  
66 // rimuovo il freno di stazionamento  
67 SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER,   
    EVENT_PARKING_BRAKES_SET, 0, SIMCONNECT_GROUP_PRIORITY_HIGHEST,   
    SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
68 SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_RUDDER_SET, 0,   
    SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
69 SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_AILERON_SET,   
    0, SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
70 SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET,   
    0, SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
71 //passo alla prossima fase  
72 flag_decollo = 2;  
73 }  
74 break;  
75  
76 //fase di rullaggio  
77 case 2:  
78 {  
79     // manetta al 90%  
80     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_THROTTLE_SET,   
        14743, SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
81  
82     //attivo l'autopilota di heading con controllo di timone  
83     double rudder = Headingtakeoff(initial_heading, UserPlane.heading);  
84     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_RUDDER_SET,   
        static_cast<DWORD>(round(rudder)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,   
        SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
85     //autopilota di controllo del pitch per tenere il velivolo a terra  
86     double elevator = Pitchhold(0.8, UserPlane.pitch, UserPlane.pitchrate);  
87     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET,   
        static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
```

```
        SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
88  
89        //comando per richiamata  
90        if (UserPlane.velocity > 300)  
91        {  
92            flag_decollo = 3;  
93            // setto l'alettone a 0 (non servirà se implemento l'ari)  
94            SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER,          ↗  
                EVENT_RUDDER_SET, 0, SIMCONNECT_GROUP_PRIORITY_HIGHEST,                ↗  
                SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
95        }  
96    }  
97    break;  
98  
99    // fase di richiamata e salita rettilinea  
100    case 3:  
101    {  
102        // manetta al 90%  
103        SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_THROTTLE_SET, ↗  
            14743, SIMCONNECT_GROUP_PRIORITY_HIGHEST, SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
104  
105        //autopilota di mantenimento dell'heading con controllo di alettone  
106        double aileron = Headinghold(initial_heading, UserPlane.heading, 10, UserPlane.bank,    ↗  
            UserPlane.rollrate);  
107        SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_AILERON_SET, ↗  
            static_cast<DWORD>(round(aileron)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,    ↗  
            SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
108  
109        //autopilota di controllo della quota con controllo dell'equilibratore  
110        double elevator = Pitchhold(-15, UserPlane.pitch, UserPlane.pitchrate);  
111        SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET, ↗  
            static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,    ↗  
            SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
```

```
112
113         //comando per virata di allineamento alla crociera
114         if (UserPlane.altitude > quota_crociera / 3)
115         {
116             flag_decollo = 4;
117         }
118     }
119     break;
120 }
121 }
122
123
124 //fase di immissione in rotta di crociera e mantenimento della quota in crociera
125 if (flag_decollo == 4)
126 {
127     // manetta al 70%
128     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_THROTTLE_SET,
129                                     static_cast<DWORD>(round(16383 * .7)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
130                                     SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
131
132     //autopilota di controllo della quota con controllo dell'equilibratore
133     if (UserPlane.altitude < quota_crociera - 250)
134     {
135         double elevator = Pitchhold(-15, UserPlane.pitch, UserPlane.pitchrate);
136         SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET,
137                                         static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
138                                         SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
139     }
140     else if (UserPlane.altitude > quota_crociera + 250)
141     {
142         double elevator = Pitchhold(15, UserPlane.pitch, UserPlane.pitchrate);
```

```
141         SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET, ↗  
            static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST, ↗  
            SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
142     }  
143  
144     else  
145     {  
146  
147         double elevator = Altitudehold(quota_crociera, UserPlane.altitude, 15, UserPlane.pitch, ↗  
            UserPlane.pitchrate);  
148         SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET, ↗  
            static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST, ↗  
            SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
149     }  
150  
151     //autopilota di mantenimento dell'heading con controllo di alettone  
152     double aileron = Headinghold(heading_crociera, UserPlane.heading, 20, UserPlane.bank, ↗  
        UserPlane.rollrate);  
153     SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_AILERON_SET, ↗  
        static_cast<DWORD>(round(aileron)), SIMCONNECT_GROUP_PRIORITY_HIGHEST, ↗  
        SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
154  
155     }  
156  
157  
158     //fase di crociera in quota a velocità ridotta  
159     if (flag_decollo == 100)  
160     {  
161  
162         std::vector<double> NED = EcefToNEDPhi(OtherPlane.latitude, OtherPlane.longitude, ↗  
            OtherPlane.altitude, UserPlane.latitude, UserPlane.longitude, UserPlane.altitude, ↗  
            OtherPlane.heading);  
163         printf("NED NORD %.2f", NED[0]);
```

```
164         printf("EAST %.2f", NED[1]);
165         printf("DOWN %.2f", NED[2]);
166
167         // autopilota di manetta
168         double throttle = ForwardSeparation( NED, -40.0, UserPlane.velocityZ, UserPlane.accelerationZ);
169         SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_THROTTLE_SET,
170                                     static_cast<DWORD>(throttle), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
171                                     SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
172
173         double elevator = VerticalSeparation(NED, 20, 20, UserPlane.pitch, UserPlane.pitchrate);
174         SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_ELEVATOR_SET,
175                                     static_cast<DWORD>(round(elevator)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
176                                     SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
177
178         double distance = sqrt(NED[1] * NED[1] + NED[0] * NED[0] + NED[2] * NED[2]);
179         printf("distance %.2f\n", distance);
180         if (distance > abs(1000))
181         {
182             //autopilota di mantenimento dell'heading con controllo di alettone
183             double aileron = LateralSeparationApproach(NED, 20, UserPlane.bank, UserPlane.heading,
184                                                         UserPlane.rollrate);
185             //double aileron = LateralSeparation(NED, 20, 35, UserPlane.bank, UserPlane.heading,
186             //                                     UserPlane.rollrate);
187             SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_AILERON_SET,
188                                     static_cast<DWORD>(round(aileron)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,
189                                     SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);
190
191         }
192         else
193         {
194             double aileron = LateralSeparation( NED, 20, 35, UserPlane.bank, UserPlane.rollrate);
195             SimConnect_TransmitClientEvent(hSimConnect, SIMCONNECT_OBJECT_ID_USER, EVENT_AILERON_SET,
```

```
static_cast<DWORD>(round(aileron)), SIMCONNECT_GROUP_PRIORITY_HIGHEST,  
SIMCONNECT_EVENT_FLAG_GROUPID_IS_PRIORITY);  
  
189  
190     }  
191  
192     }  
193  
194  
195     }  
196     //definisco la frequenza dell'add-on  
197     Sleep(50);  
198 }  
199  
200 //chiusura applicazione e file output  
201 hr = SimConnect_Close(hSimConnect);  
202  
203 printf("\nDisconnected from Prepar3D ");  
204 system("pause");  
205 }  
206 else  
207 {  
208     printf("\nFailed to Connect to Prepar3D ");  
209     system("pause");  
210 }  
211 system("pause");  
212 return 0;  
213 }  
214
```