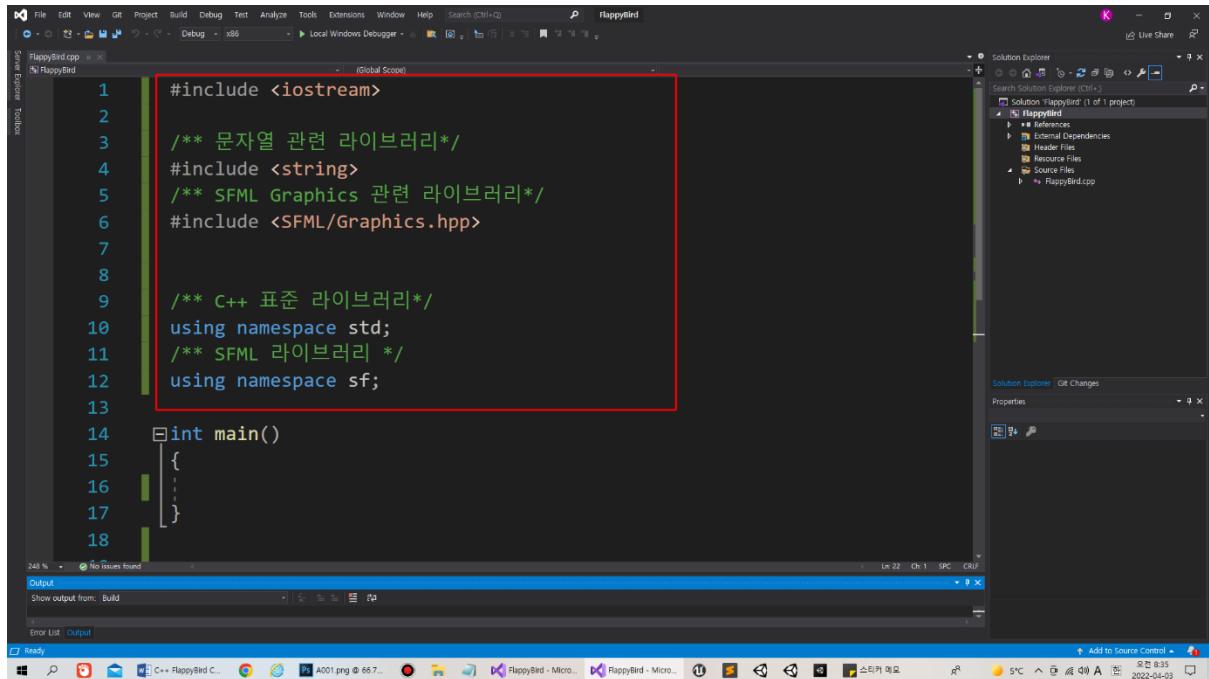


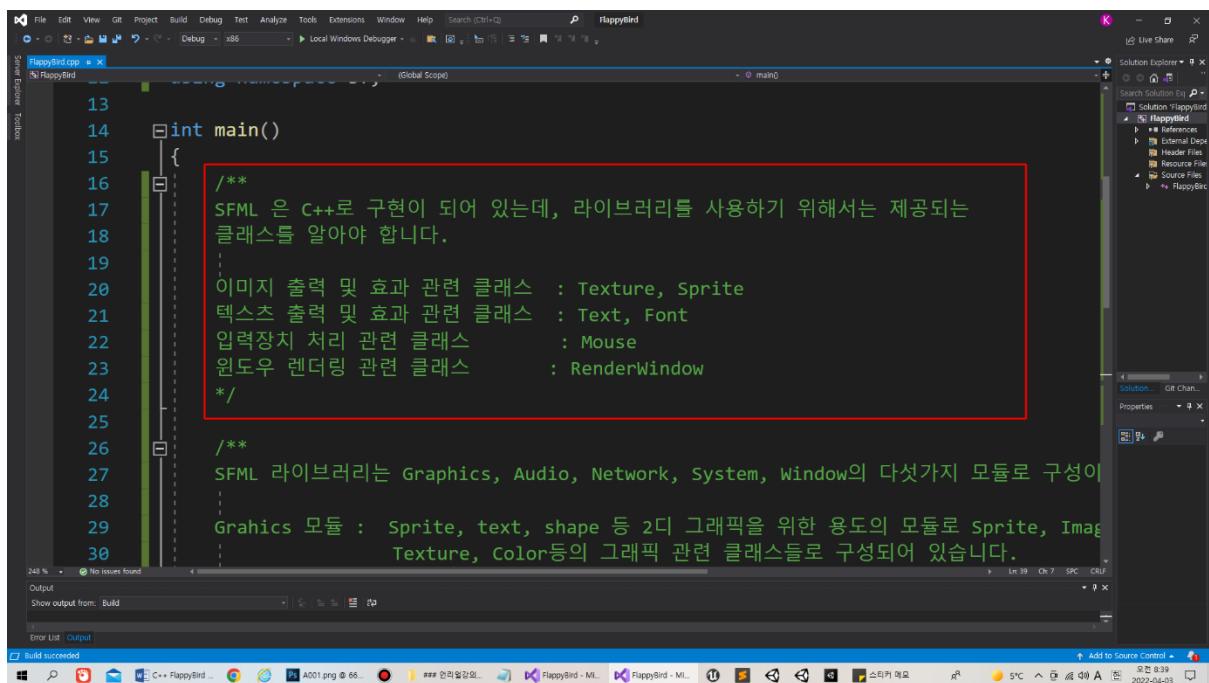
이번시간에는 Flappy bird 게임의 상태마신에 대해서 알아보도록 합니다.

FlappyBird 구현부에서 필요한 라이브러리를 가져옵니다.



```
#include <iostream>
/** 문자열 관련 라이브러리*/
#include <string>
/** SFML Graphics 관련 라이브러리*/
#include <SFML/Graphics.hpp>
/** C++ 표준 라이브러리*/
using namespace std;
/** SFML 라이브러리 */
using namespace sf;

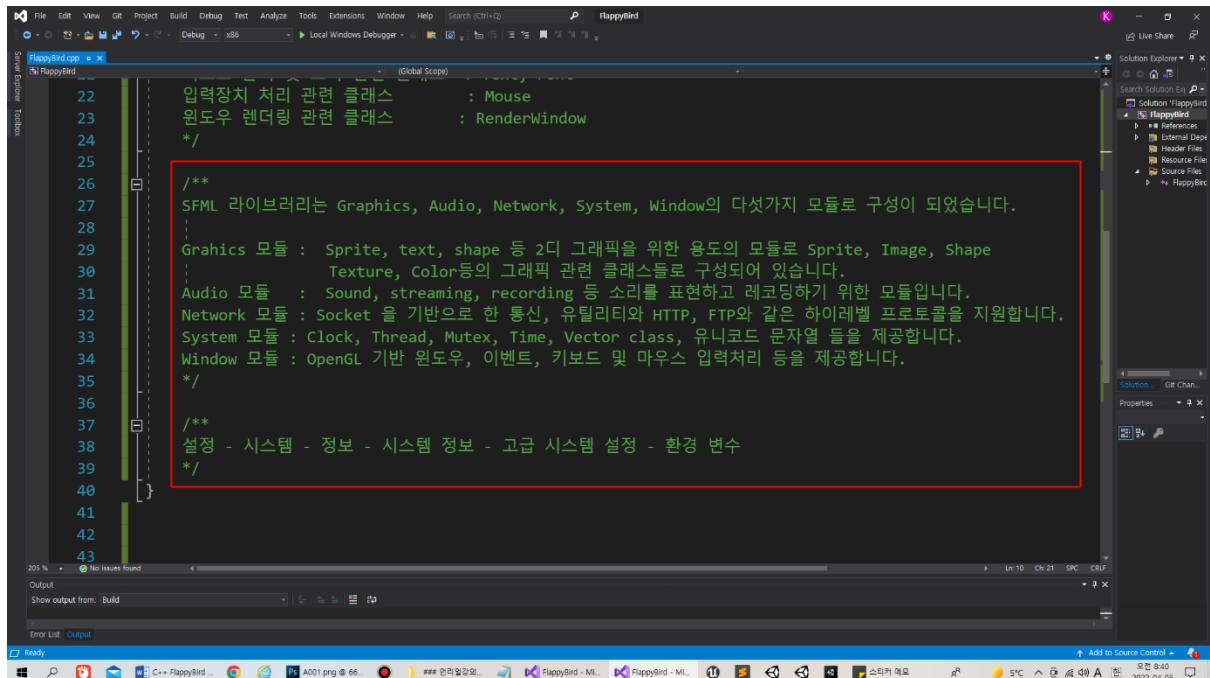
int main()
```



```
/*
SFML 은 C++로 구현이 되어 있는데, 라이브러리를 사용하기 위해서는 제공되는
클래스를 알아야 합니다.

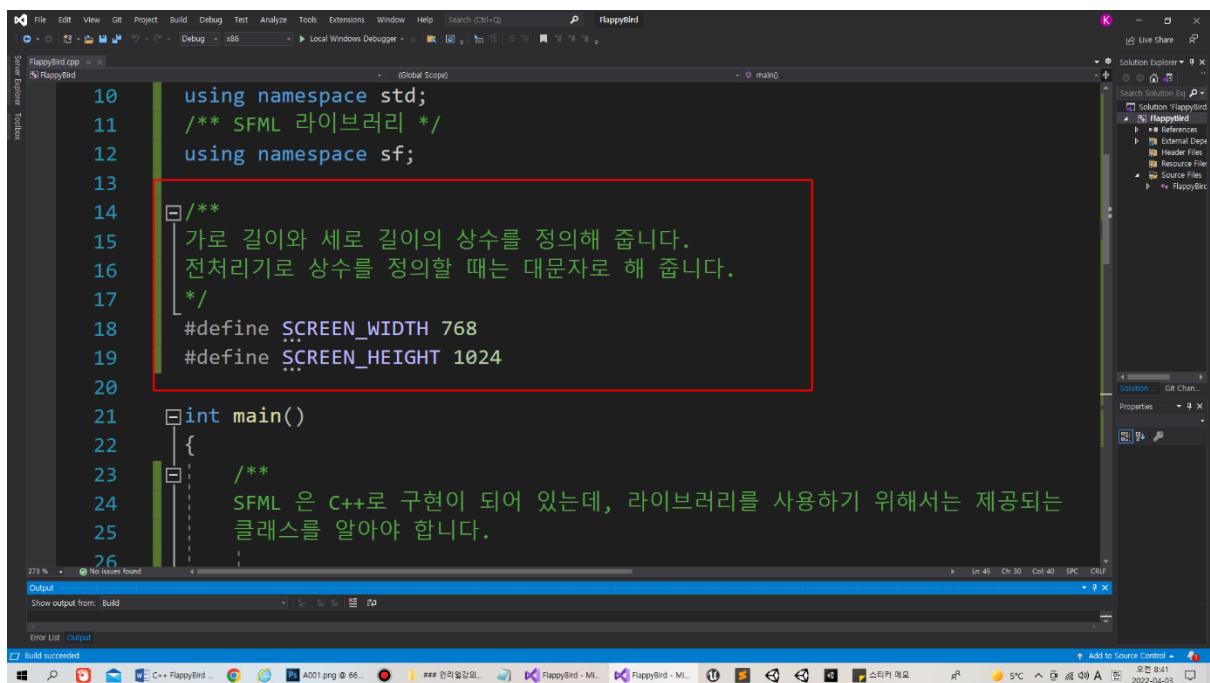
이미지 출력 및 효과 관련 클래스 : Texture, Sprite
텍스트 출력 및 효과 관련 클래스 : Text, Font
입력장치 처리 관련 클래스 : Mouse
윈도우 렌더링 관련 클래스 : RenderWindow
*/

/*
SFML 라이브러리는 Graphics, Audio, Network, System, Window의 다섯가지 모듈로 구성이
Graphics 모듈 : Sprite, text, shape 등 2d 그래픽을 위한 용도의 모듈로 Sprite, Image,
Texture, Color등의 그래픽 관련 클래스들로 구성되어 있습니다.
*/
```



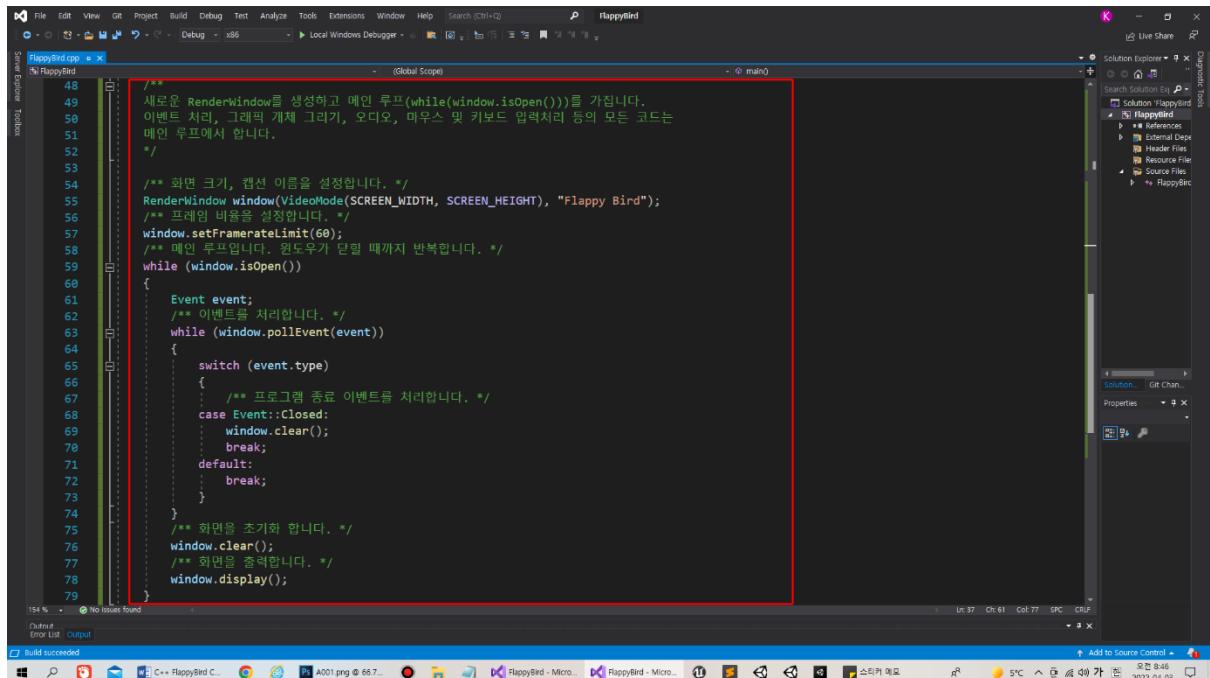
```
22     입력장치 처리 관련 클래스      : Mouse
23     윈도우 렌더링 관련 클래스      : RenderWindow
24
25
26     /**
27      SFML 라이브러리는 Graphics, Audio, Network, System, Window의 다섯가지 모듈로 구성이 되었습니다.
28
29      Graphics 모듈 : Sprite, text, shape 등 2D 그래픽을 위한 용도의 모듈로 sprite, Image, Shape
30          Texture, Color등의 그래픽 관련 클래스들로 구성되어 있습니다.
31      Audio 모듈   : Sound, streaming, recording 등 소리를 표현하고 레코딩하기 위한 모듈입니다.
32      Network 모듈 : Socket을 기반으로 한 통신, 유트리티와 HTTP, FTP와 같은 하이레벨 프로토콜을 지원합니다.
33      System 모듈  : Clock, Thread, Mutex, Time, Vector class, 유니코드 문자열 등을 제공합니다.
34      Window 모듈  : OpenGL 기반 윈도우, 이벤트, 키보드 및 마우스 입력처리 등을 제공합니다.
35
36
37     /**
38      설정 - 시스템 - 정보 - 시스템 정보 - 고급 시스템 설정 - 환경 변수
39
40
41
42
43 }
```

게임화면의 가로와 세로 길이를 정의해 줍니다.



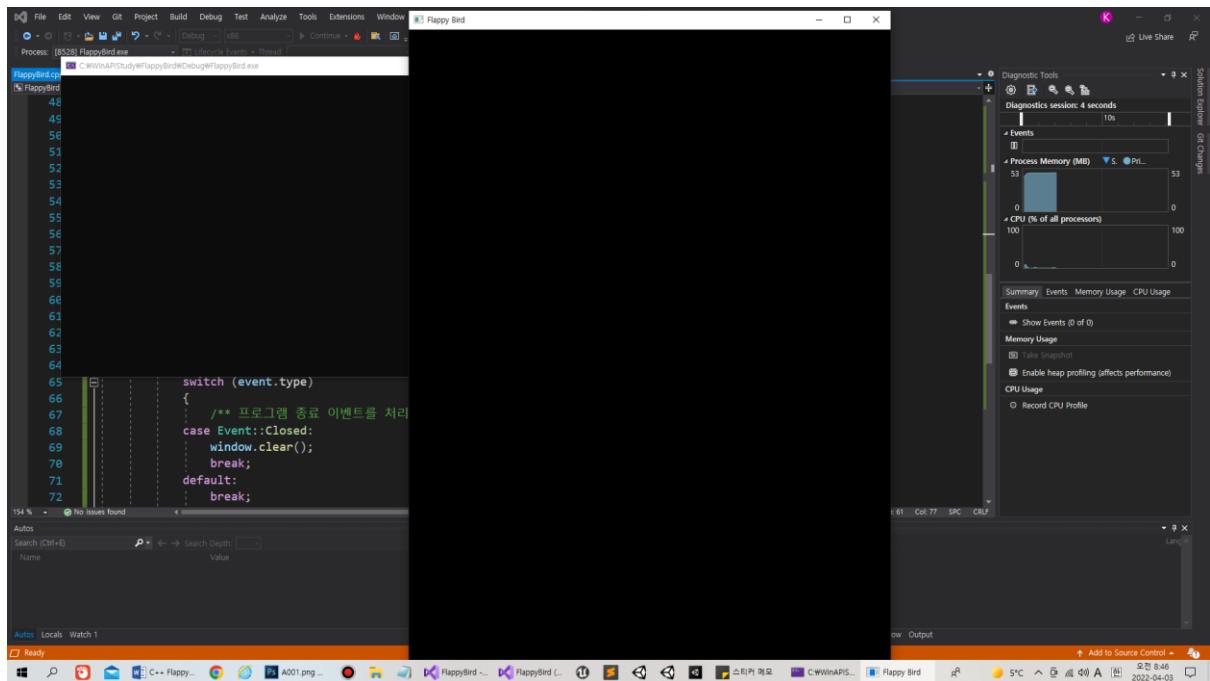
```
10     using namespace std;
11     /** SFML 라이브러리 */
12     using namespace sf;
13
14     /**
15      가로 길이와 세로 길이의 상수를 정의해 줍니다.
16      전처리기로 상수를 정의할 때는 대문자로 해 줍니다.
17
18     #define SCREEN_WIDTH 768
19     #define SCREEN_HEIGHT 1024
20
21     int main()
22     {
23
24         /**
25          SFML은 C++로 구현이 되어 있는데, 라이브러리를 사용하기 위해서는 제공되는
26          클래스를 알아야 합니다.
```

기본적인 화면 구성을 해주도록 합니다.



```
48
49     /**
50      * 새로운 RenderWindow를 생성하고 메인 루프(while(window.isOpen()))를 가집니다.
51      * 이벤트 처리, 그래픽 캐치 그리기, 오디오, 마우스 및 키보드 입력처리 등의 모든 코드는
52      * 메인 루프에서 합니다.
53      */
54
55     /** 화면 크기, 캡션 이름을 설정합니다. */
56     RenderWindow window(VideoMode(SCREEN_WIDTH, SCREEN_HEIGHT), "Flappy Bird");
57     /** 프레임 비율을 설정합니다. */
58     window.setFramerateLimit(60);
59     /** 메인 루프입니다. 원도우가 닫힐 때까지 반복합니다. */
60     while (window.isOpen())
61     {
62         Event event;
63         /** 이벤트를 처리합니다. */
64         while (window.pollEvent(event))
65         {
66             switch (event.type)
67             {
68                 /** 프로그램 종료 이벤트를 처리합니다. */
69                 case Event::Closed:
70                     window.clear();
71                     break;
72                     default:
73                     break;
74             }
75             /** 화면을 초기화 합니다. */
76             window.clear();
77             /** 화면을 출력합니다. */
78             window.display();
79         }
80     }
81 }
```

결과를 확인해 봅니다.

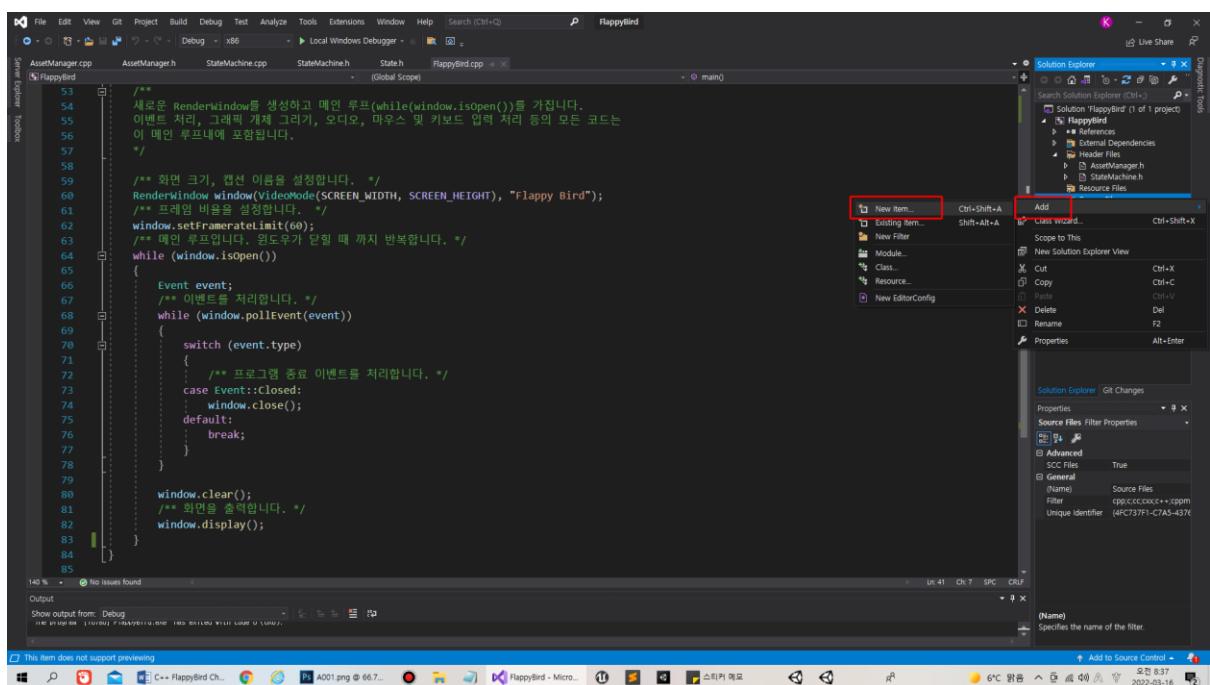
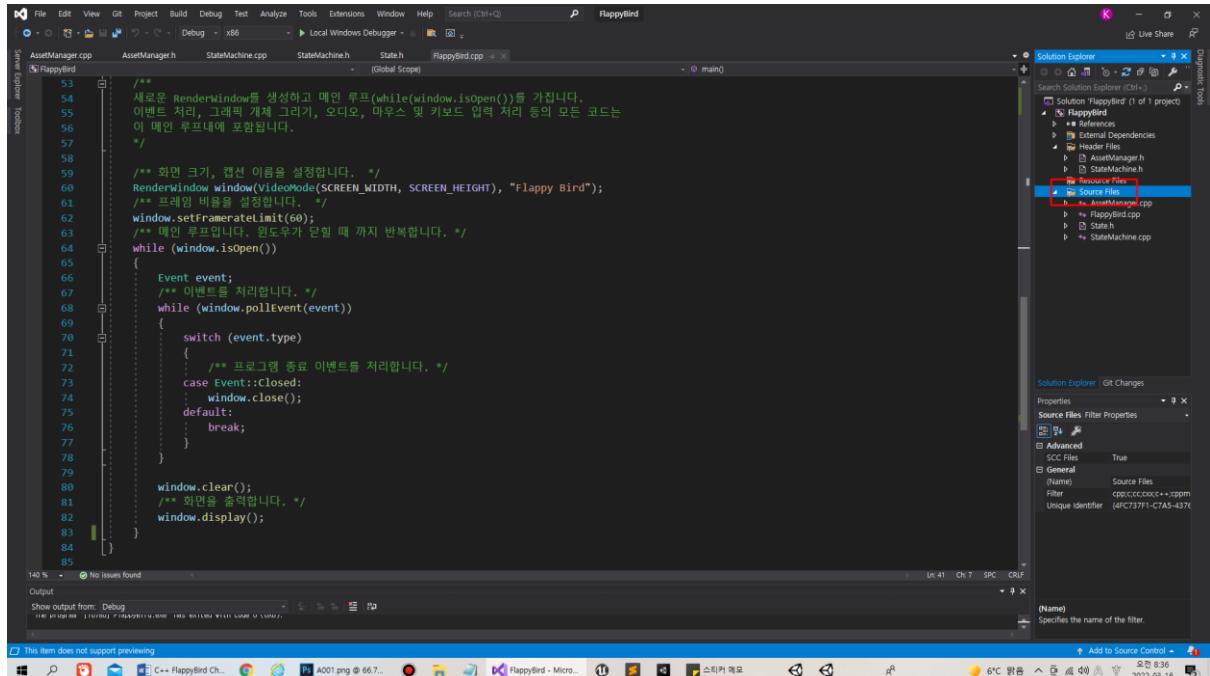


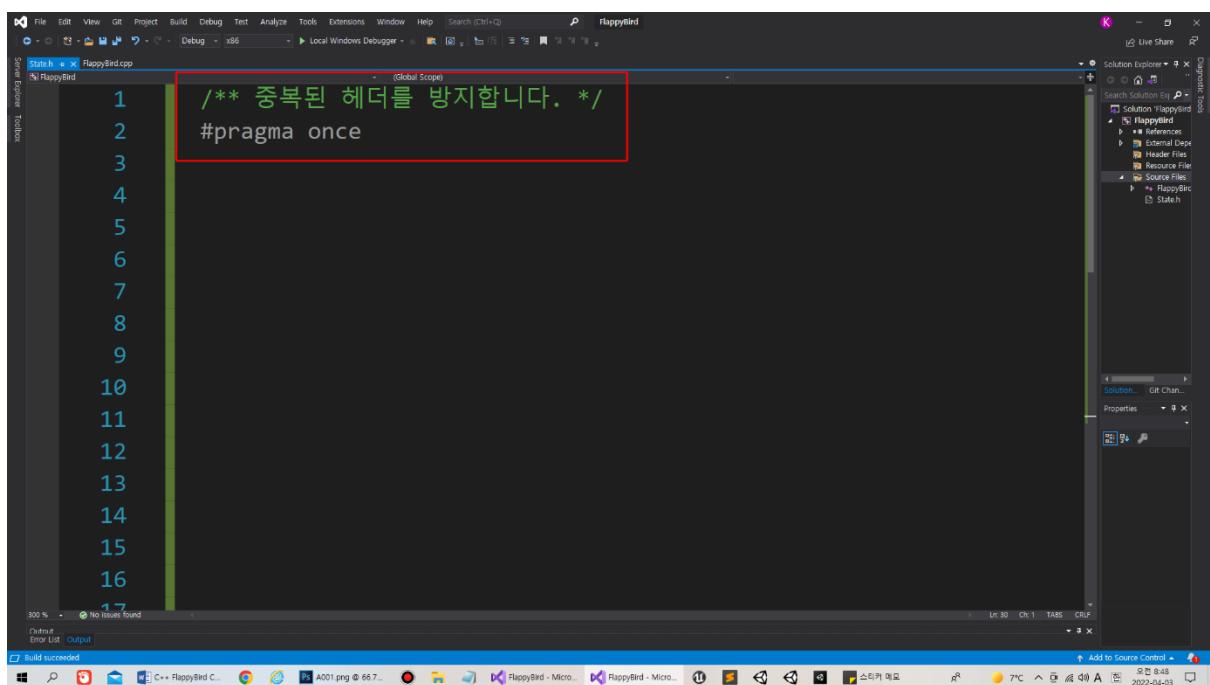
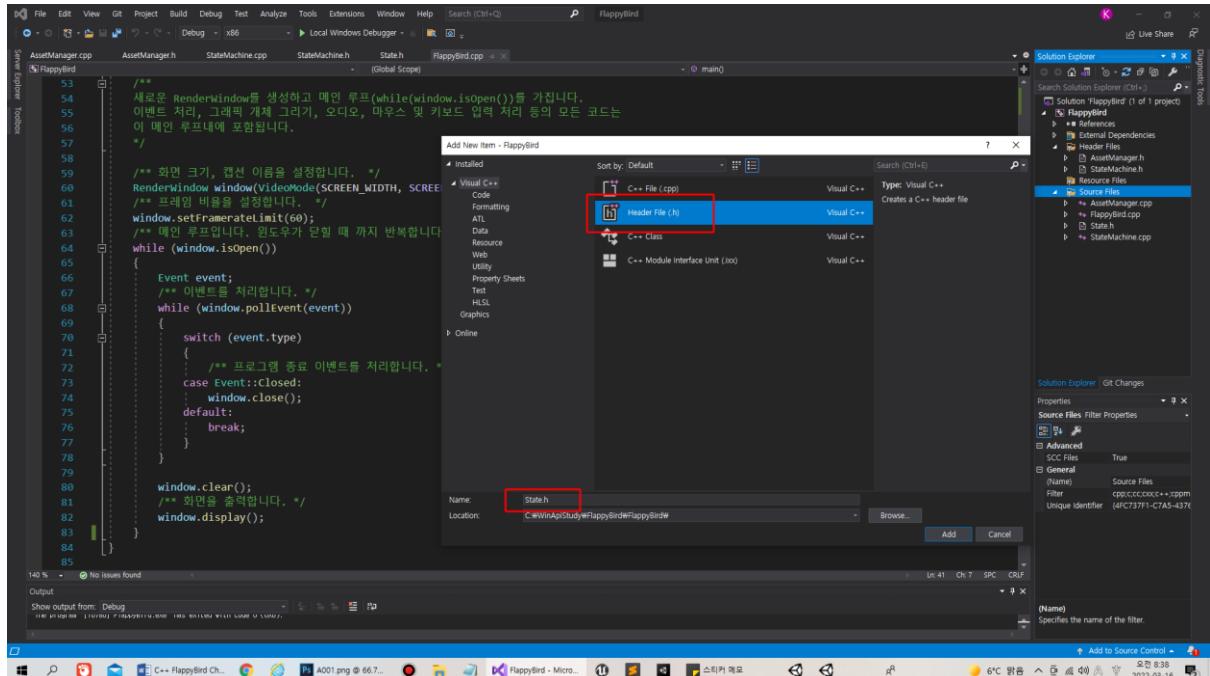
Process: (6528) FlappyBird.exe

FlappyBird

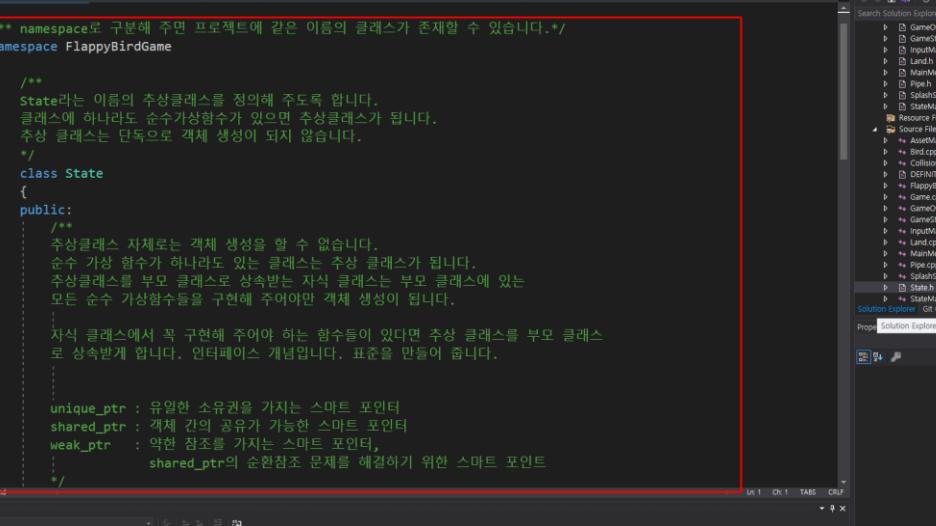
```
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65     switch (event.type)
66     {
67         /** 프로그램 종료 이벤트를 처리
68         case Event::Closed:
69             window.clear();
70             break;
71             default:
72             break;
73     }
74
75     /** 화면을 초기화 합니다. */
76     window.clear();
77     /** 화면을 출력합니다. */
78     window.display();
79 }
```

State라는 이름으로 헤더파일을 생성해 줍니다.



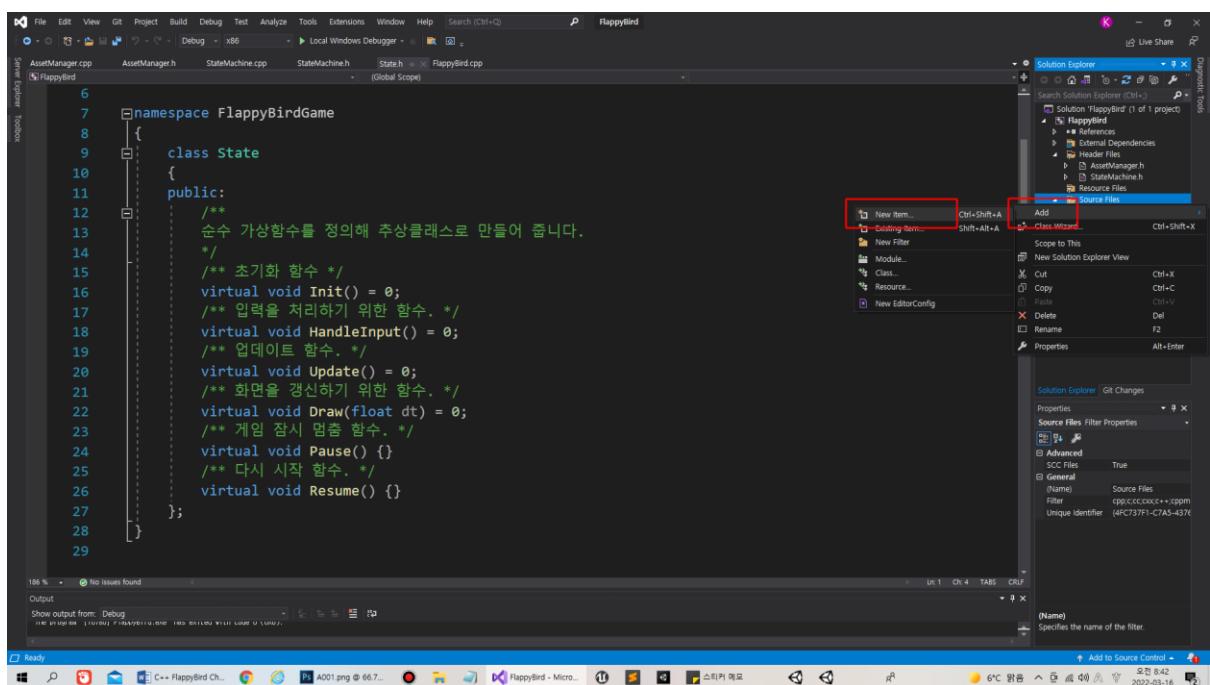
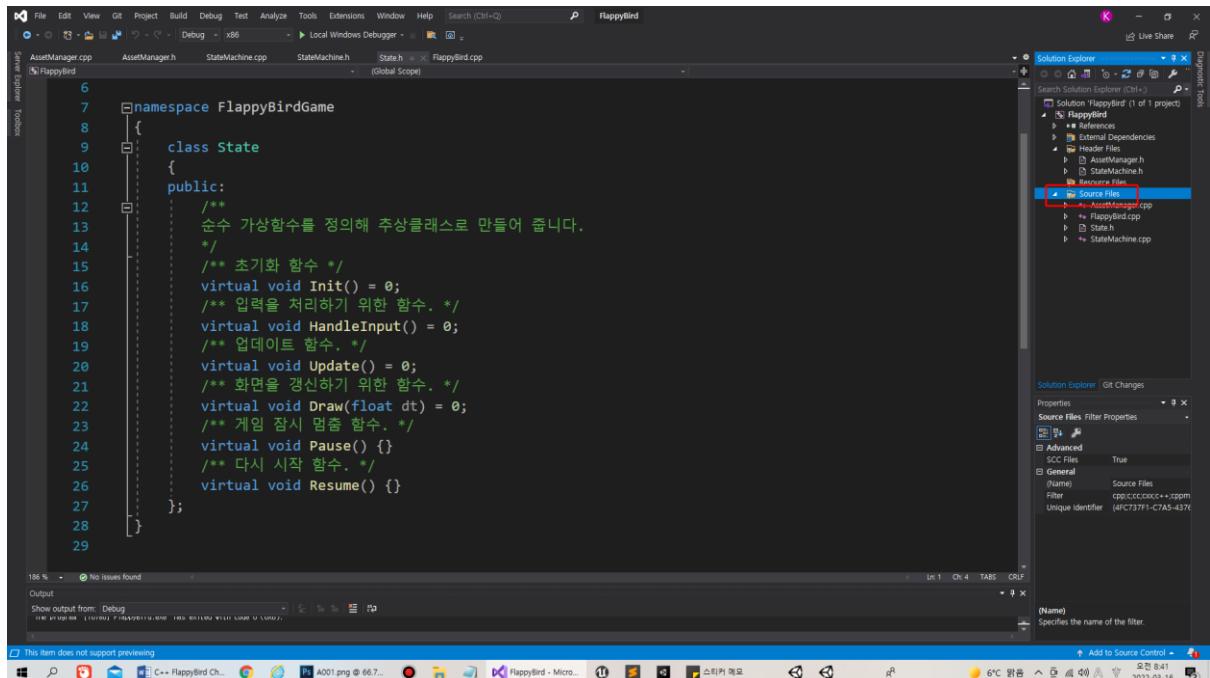


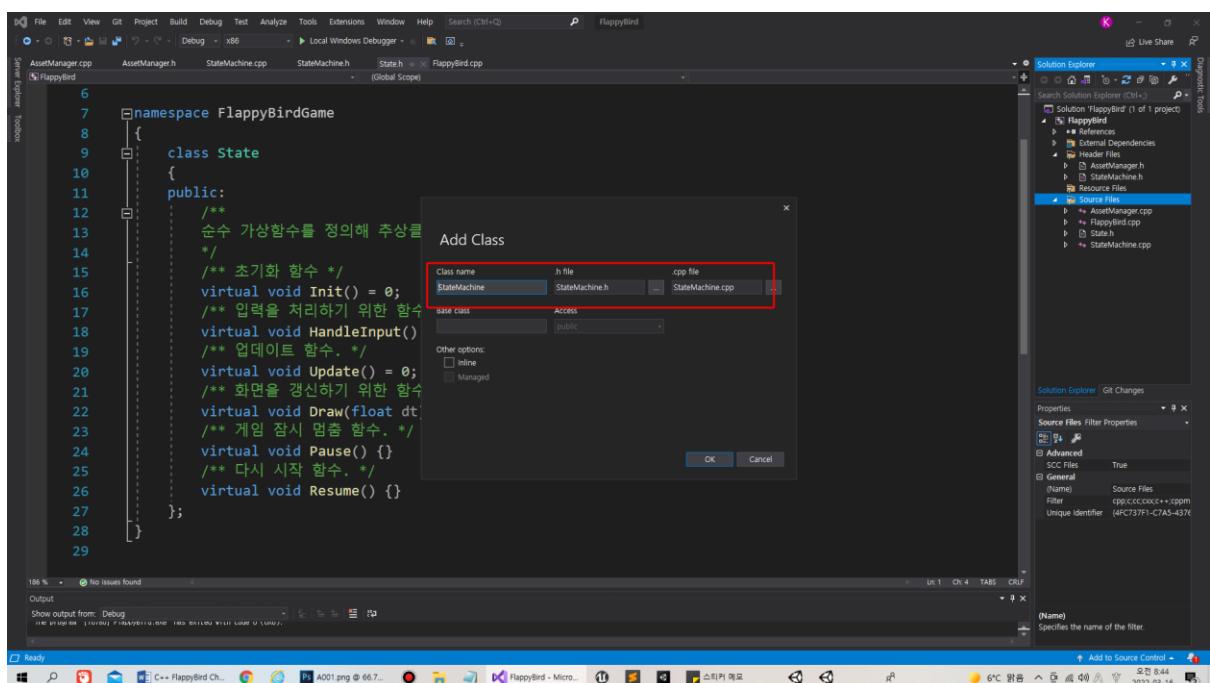
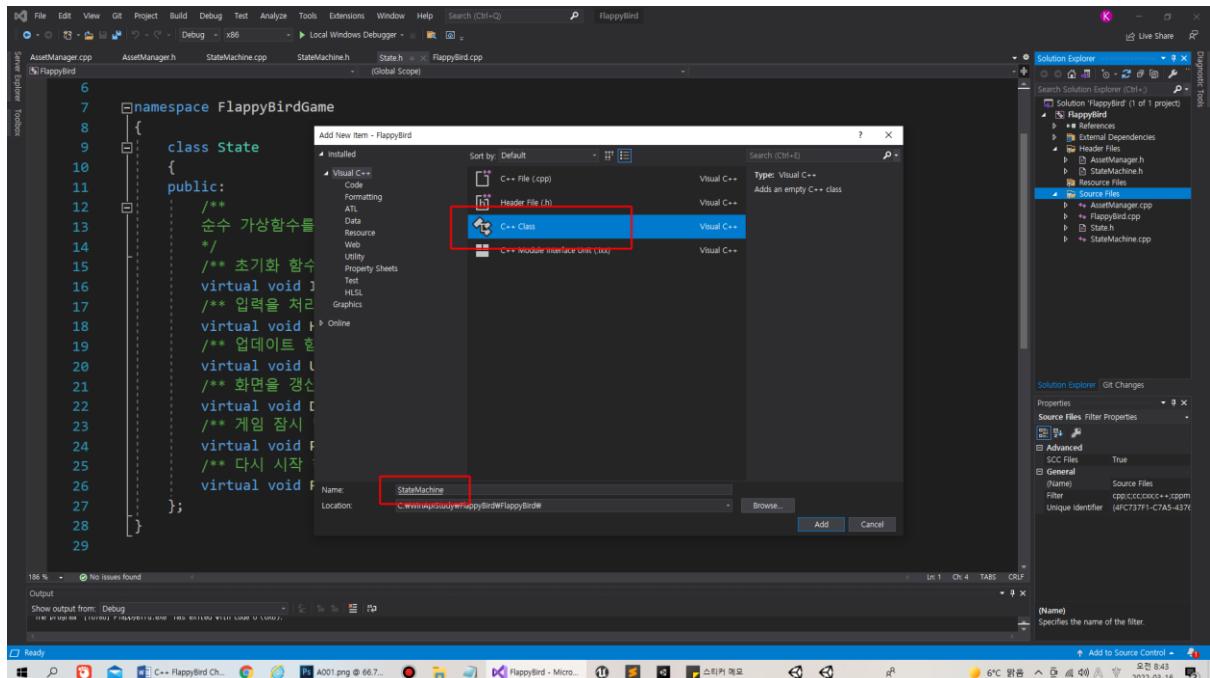
State라는 이름의 클래스를 만들어 주도록 합니다.



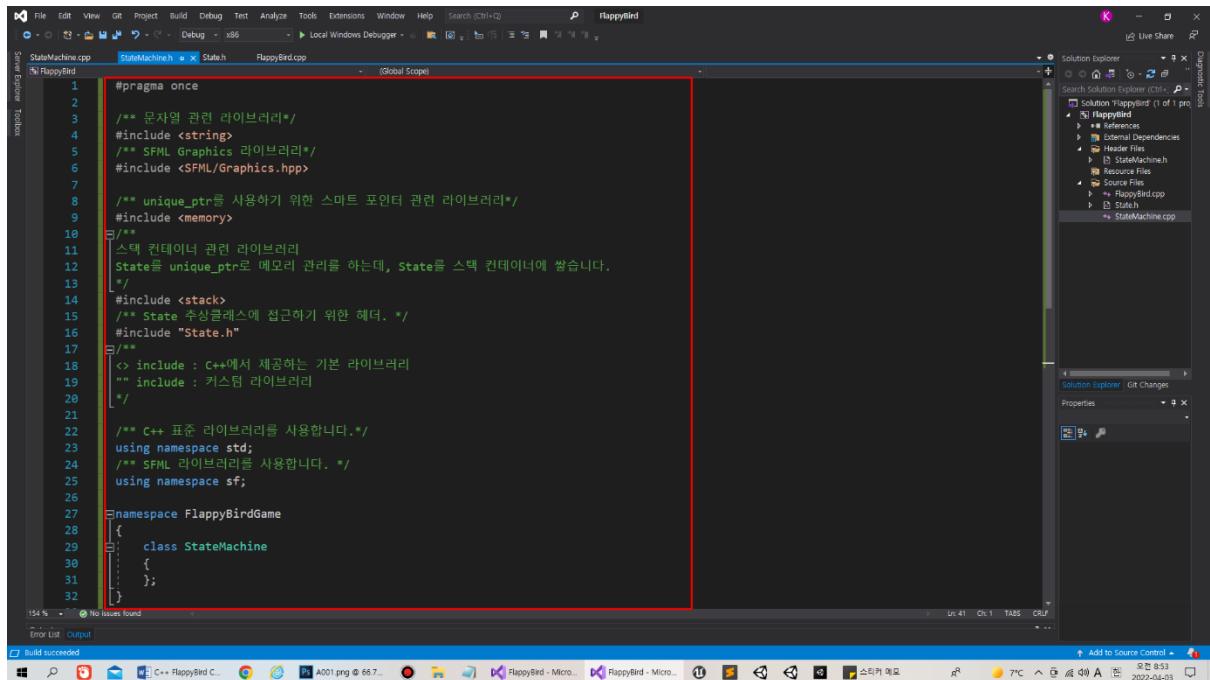
```
3 //** namespace 구분해 주면 프로젝트에 같은 이름의 클래스가 존재할 수 있습니다.*/
4
5 #ifndef STATE_H
6 #define STATE_H
7
8 #include "Game.h"
9
10 #include "InputManager.h"
11
12 #include "State.h"
13
14 class State
15 {
16 public:
17     /**
18      * 주상클래스 자체로는 객체 생성을 할 수 없습니다.
19      * 순수 가상 함수가 하나라도 있는 클래스는 주상 클래스가 됩니다.
20      * 주상클래스를 부모 클래스로 상속받는 자식 클래스는 부모 클래스에 있는
21      * 모든 순수 가상함수들을 구현해 주어야만 객체 생성이 됩니다.
22      */
23
24     unique_ptr : 유일한 소유권을 가지는 스마트 포인터
25     shared_ptr : 객체 간의 공유가 가능한 스마트 포인터
26     weak_ptr : 약한 참조를 가지는 스마트 포인터,
27                 shared_ptr의 순환참조 문제를 해결하기 위한 스마트 포인트
28
29 };
```

이제 StateMachine이라는 이름의 클래스를 만들어 주도록 합니다.

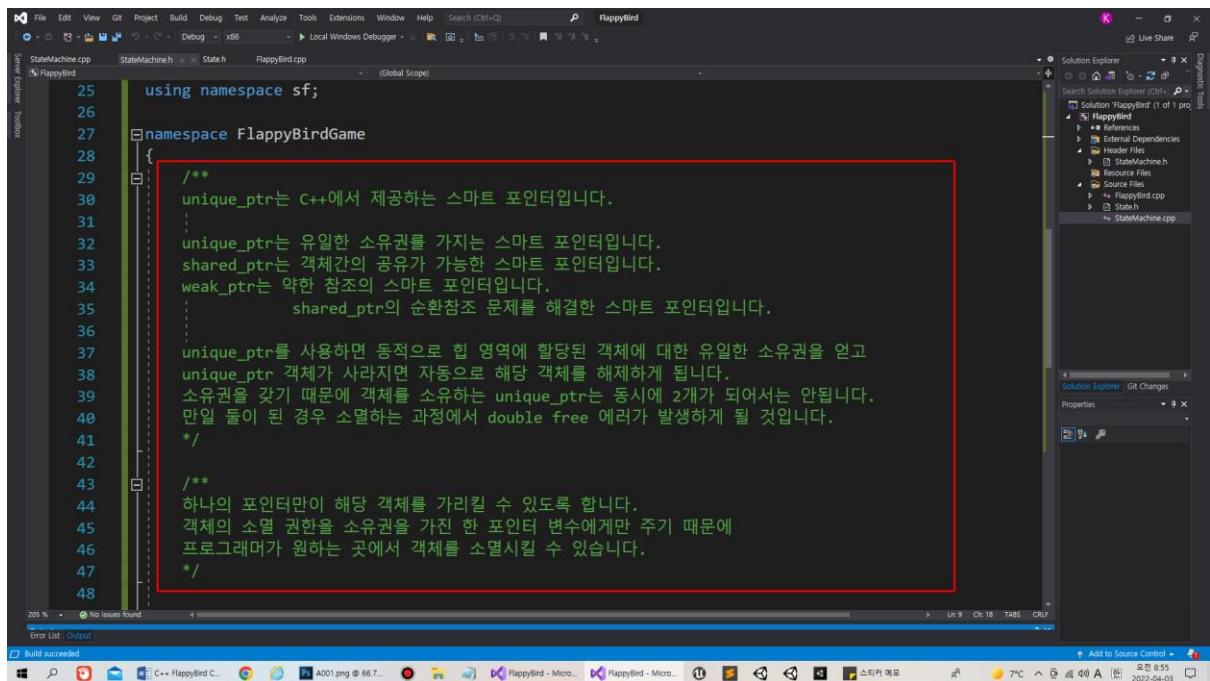




헤더 파일에서 필요한 라이브러리를 추가해 주도록 합니다.

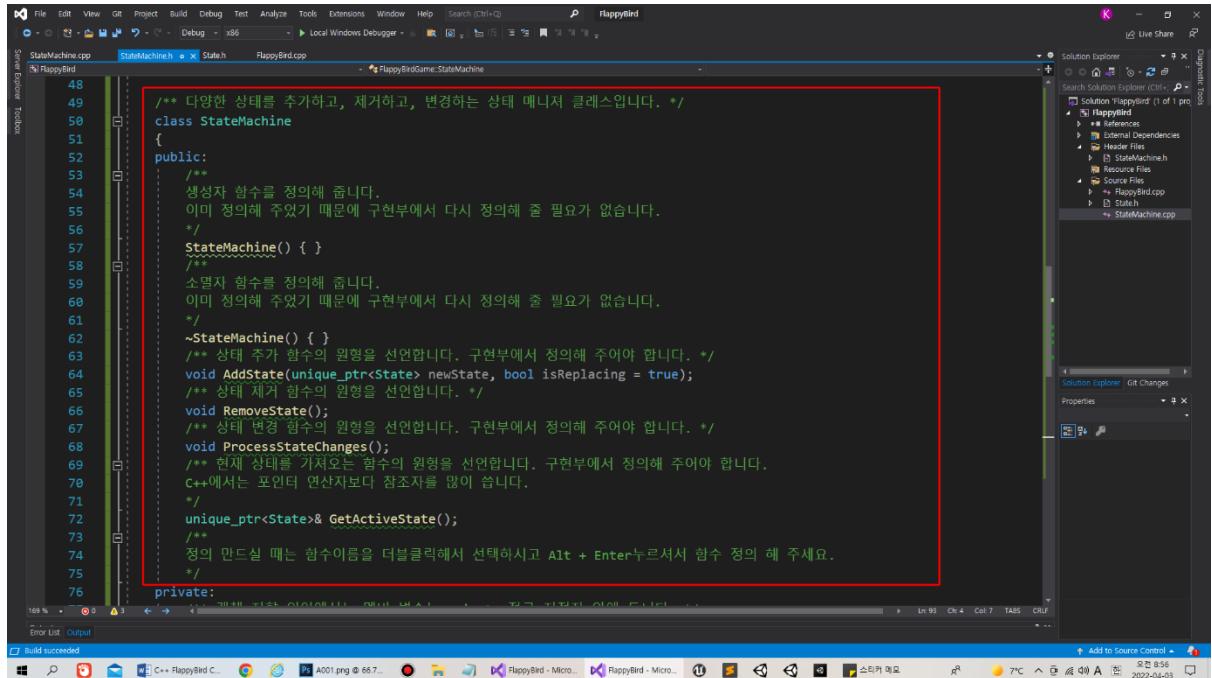


```
1 #pragma once
2
3 /** 문자열 관련 라이브러리 */
4 #include <string>
5 /** SFML Graphics 라이브러리 */
6 #include <SFML/Graphics.hpp>
7
8 /** unique_ptr를 사용하기 위한 스마트 포인터 관련 라이브러리 */
9 #include <memory>
10
11 /**
12  * 스택 컨테이너 관련 라이브러리
13  * State를 unique_ptr로 메모리 관리를 하는데, State를 스택 컨테이너에 쓸습니다.
14  */
15 #include <stack>
16 /**
17  * State 주상클래스에 접근하기 위한 헤더.
18  */
19 #include "State.h"
20
21 /**
22  * <> include : C++에서 제공하는 기본 라이브러리
23  * "" include : 커스텀 라이브러리
24  */
25
26 /**
27  * C++ 표준 라이브러리를 사용합니다.
28  */
29 using namespace std;
30 /**
31  * SFML 라이브러리를 사용합니다.
32  */
33 using namespace sf;
34
35
36 namespace FlappyBirdGame
37 {
38     class StateMachine
39     {
40     ...
41     };
42 }
```

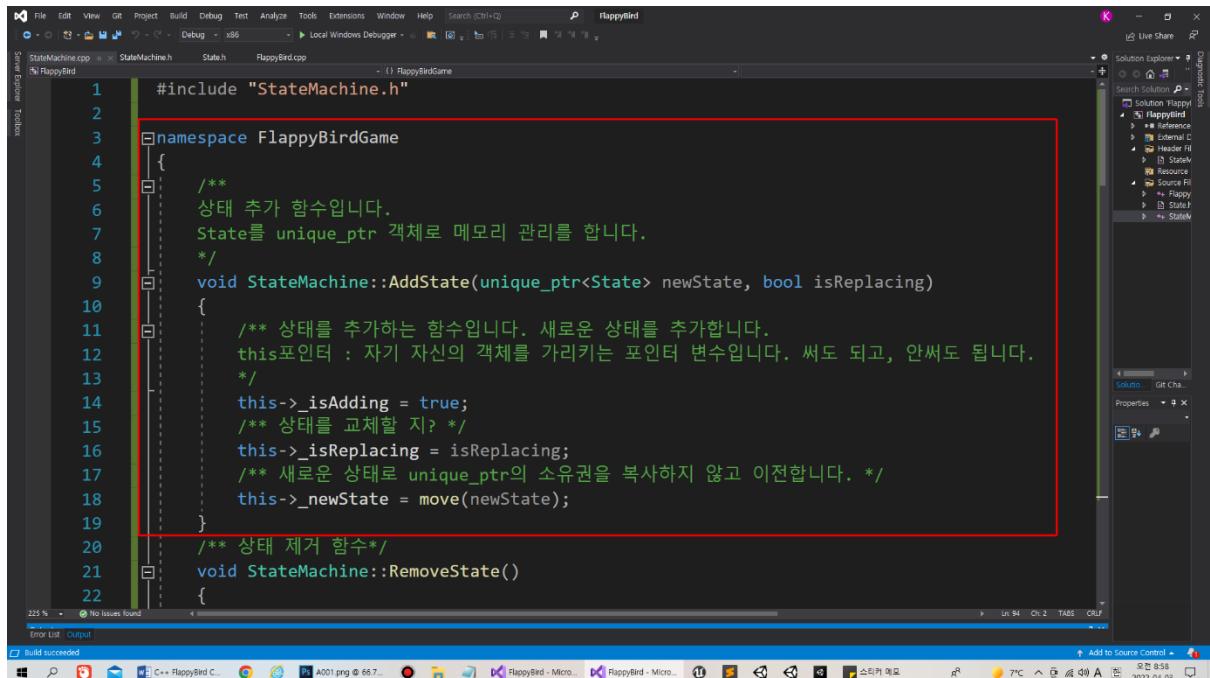


```
25     using namespace sf;
26
27     namespace FlappyBirdGame
28     {
29         /**
30          * unique_ptr는 C++에서 제공하는 스마트 포인터입니다.
31
32          * unique_ptr는 유일한 소유권을 가지는 스마트 포인터입니다.
33          * shared_ptr는 객체간의 공유가 가능한 스마트 포인터입니다.
34          * weak_ptr는 약한 참조의 스마트 포인터입니다.
35          * shared_ptr의 순환참조 문제를 해결한 스마트 포인터입니다.
36
37          * unique_ptr를 사용하면 동적으로 힙 영역에 할당된 객체에 대한 유일한 소유권을 얻고
38          * unique_ptr 객체가 사라지면 자동으로 해당 객체를 해제하게 됩니다.
39          * 소유권을 갖기 때문에 객체를 소유하는 unique_ptr는 동시에 2개가 되어서는 안됩니다.
40          * 만일 둘이 된 경우 소멸하는 과정에서 double free 에러가 발생하게 될 것입니다.
41          */
42
43         /**
44             하나의 포인터만이 해당 객체를 가리킬 수 있도록 합니다.
45             객체의 소멸 권한을 소유권을 가진 한 포인터 변수에게만 주기 때문에
46             프로그래머가 원하는 곳에서 객체를 소멸시킬 수 있습니다.
47         */
48     }
```

필요한 함수의 원형을 선언해 줍니다.

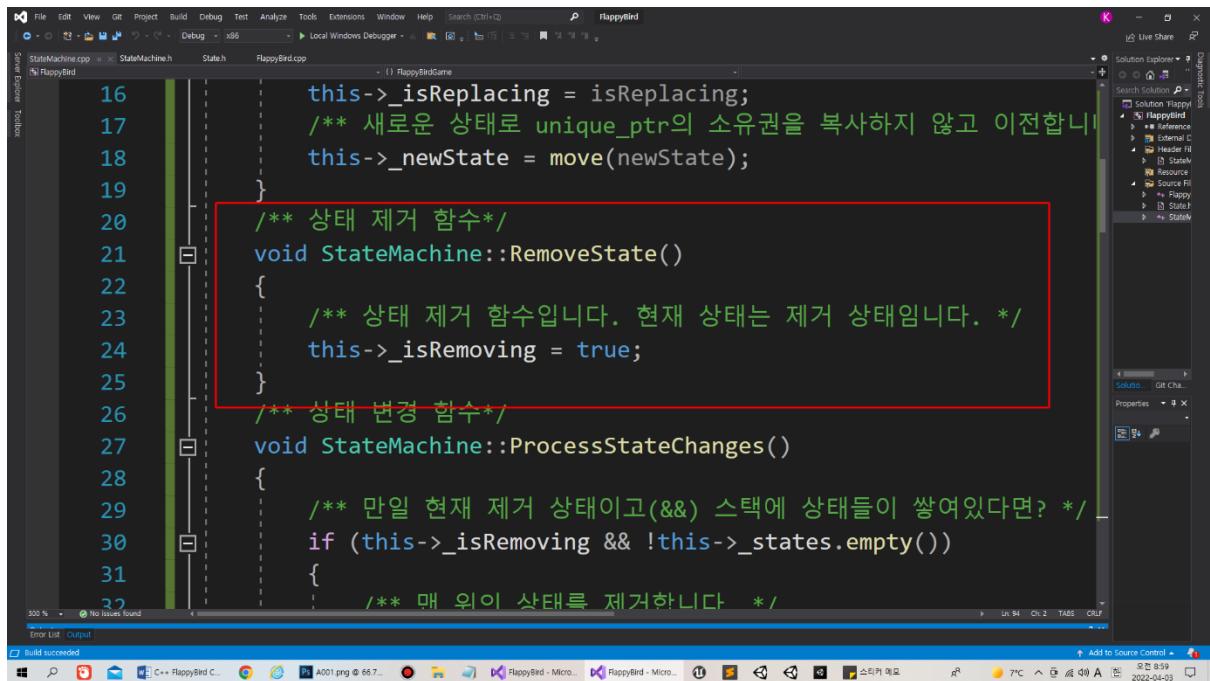


```
48
49  /** 다양한 상태를 추가하고, 제거하고, 변경하는 상태 매니저 클래스입니다. */
50  class StateMachine
51  {
52  public:
53  /**
54   * 생성자 함수를 정의해 줍니다.
55   * 이미 정의해 주었기 때문에 구현부에서 다시 정의해 줄 필요가 없습니다.
56   */
57  StateMachine() { }
58  /**
59   * 소멸자 함수를 정의해 줍니다.
60   * 이미 정의해 주었기 때문에 구현부에서 다시 정의해 줄 필요가 없습니다.
61   */
62  ~StateMachine() { }
63  /**
64   * 상태 추가 함수의 원형을 선언합니다. 구현부에서 정의해 주어야 합니다.
65   * void AddState(unique_ptr<State> newState, bool isReplacing = true);
66  /**
67   * 상태 제거 함수의 원형을 선언합니다. */
68  void RemoveState();
69  /**
70   * 상태 변경 함수의 원형을 선언합니다. 구현부에서 정의해 주어야 합니다.
71   * void ProcessStateChanges();
72  /**
73   * 현재 상태를 가져오는 함수의 원형을 선언합니다. 구현부에서 정의해 주어야 합니다.
74   * C++에서는 포인터 연산자보다 참조자를 많이 씁니다.
75  /**
76   * 정의 만드실 때는 함수 이름을 더블클릭해서 선택하시고 Alt + Enter 누르면서 함수 정의 해 주세요.
77  */
78  private:
79  /**
80   * 객체 지향 언어에서는 멤버 변수는 private 접근 지정자 안에 둡니다.
81   */
82  /**
83   * 상태들을 스택 컨테이너에 쌓아 놓기 위한 스택 컨테이너
84   * 스택은 쌓는 거라서 맨 나중에 쌓인게 맨 위에 있습니다.
85   * top() 함수로 맨 위에 있는 요소를 반환합니다.
86   * pop() 함수로 맨 위에 있는 요소를 제거합니다.
87  */
88  stack<unique_ptr<State>> _states;
89  /**
90   * 새 상태 */
91  unique_ptr<State> _newState;
92  /**
93   * 상태를 없앨 것인지? */
94  bool _isReplacing;
95  /**
96   * 상태를 추가할 것인지? */
97  bool _isAdding;
98  /**
99   * 상태를 교체할 것인지? */
100 bool _isRemoving;
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2
```



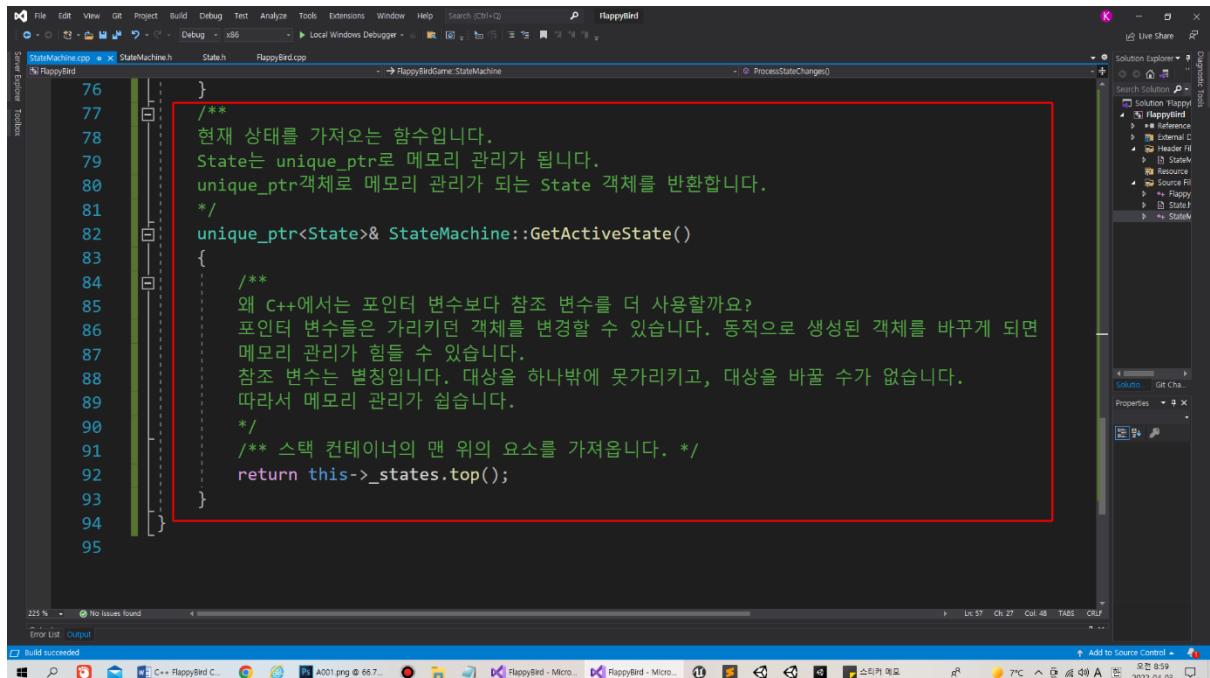
```
1 #include "StateMachine.h"
2
3 namespace FlappyBirdGame
4 {
5     /**
6      * 상태 추가 함수입니다.
7      * State를 unique_ptr 객체로 메모리 관리를 합니다.
8      */
9     void StateMachine::AddState(unique_ptr<State> newState, bool isReplacing)
10    {
11        /**
12         * 상태를 추가하는 함수입니다. 새로운 상태를 추가합니다.
13         * this포인터 : 자기 자신의 객체를 가리키는 포인터 변수입니다. 써도 되고, 안써도 됩니다.
14         */
15        this->_isAdding = true;
16        /**
17         * 상태를 교체할 지? */
18        this->_isReplacing = isReplacing;
19        /**
20         * 새로운 상태로 unique_ptr의 소유권을 복사하지 않고 이전합니다.
21         */
22        this->_newState = move(newState);
23    }
24    /**
25     * 상태 제거 함수*/
26    void StateMachine::RemoveState()
27    {
28    }
```

RemoveState() 함수를 구현해 줍니다.



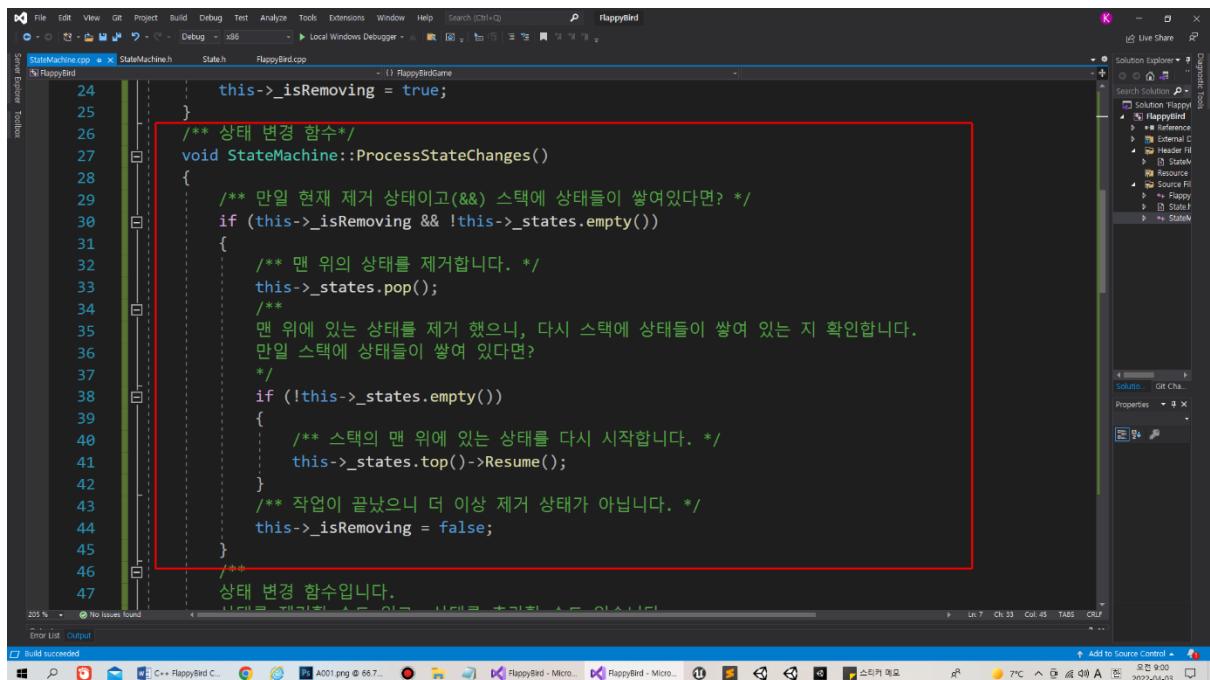
```
16     this->_isReplacing = isReplacing;
17     /**
18      * 새로운 상태로 unique_ptr의 소유권을 복사하지 않고 이전합니다.
19      */
20     /**
21      * 상태 제거 함수*/
22     void StateMachine::RemoveState()
23     {
24         /**
25          * 상태 제거 함수입니다. 현재 상태는 제거 상태입니다. */
26         this->_isRemoving = true;
27     }
28     /**
29      * 상태 변경 함수*/
30     void StateMachine::ProcessStateChanges()
31     {
32         /**
33          * 만일 현재 제거 상태이고(&&) 스택에 상태들이 쌓여있다면? */
34         if (this->_isRemoving && !this->_states.empty())
35         {
36             /**
37              * 매 위의 상태를 제거합니다 */
38         }
39     }
```

GetActiveState() 함수를 구현해 줍니다.

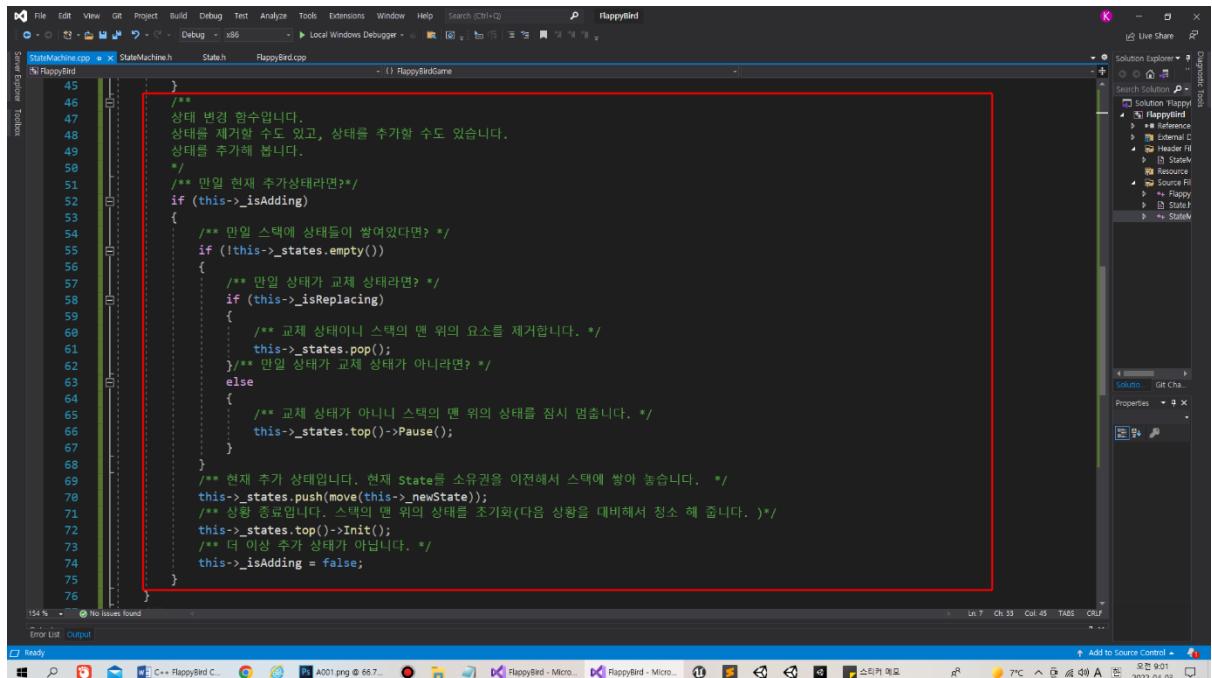


```
76 }  
77 /*  
78 현재 상태를 가져오는 함수입니다.  
79 State는 unique_ptr로 메모리 관리가 됩니다.  
80 unique_ptr 객체로 메모리 관리가 되는 State 객체를 반환합니다.  
81 */  
82 unique_ptr<State>& StateMachine::GetActiveState()  
{  
83     /*  
84     왜 C++에서는 포인터 변수보다 참조 변수를 더 사용할까요?  
85     포인터 변수들은 가리키던 객체를 변경할 수 있습니다. 동적으로 생성된 객체를 바꾸게 되면  
86     메모리 관리가 힘들 수 있습니다.  
87     참조 변수는 별정입니다. 대상을 하나밖에 못가리키고, 대상을 바꿀 수가 없습니다.  
88     따라서 메모리 관리가 쉽습니다.  
89     */  
90     /* 스택 컨테이너의 맨 위의 요소를 가져옵니다. */  
91     return this->_states.top();  
92 }  
93  
94 }  
95 }
```

ProcessStateChanges() 함수를 구현해 주도록 합니다.



```
24     this->_isRemoving = true;  
25 }  
26 /* 상태 변경 함수/  
27 void StateMachine::ProcessStateChanges()  
28 {  
29     /* 만일 현재 제거 상태이고(&&) 스택에 상태들이 쌓여있다면? */  
30     if (this->_isRemoving && !this->_states.empty())  
31     {  
32         /* 맨 위의 상태를 제거합니다. */  
33         this->_states.pop();  
34         /*  
35         맨 위에 있는 상태를 제거 했으니, 다시 스택에 상태들이 쌓여 있는지 확인합니다.  
36         만일 스택에 상태들이 쌓여 있다면?  
37         */  
38         if (!this->_states.empty())  
39         {  
40             /* 스택의 맨 위에 있는 상태를 다시 시작합니다. */  
41             this->_states.top()->Resume();  
42         }  
43         /* 작업이 끝났으니 더 이상 제거 상태가 아닙니다. */  
44         this->_isRemoving = false;  
45     }  
46     /*  
47     상태 변경 함수입니다.  
48 }
```

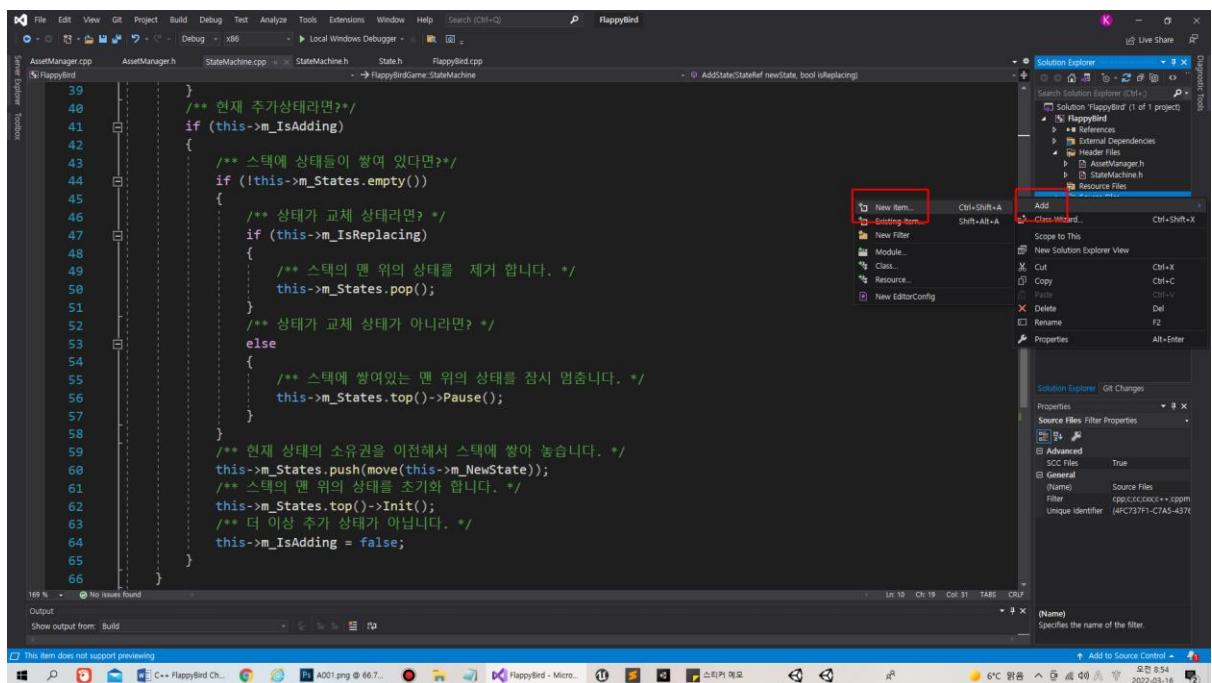


```

45     }
46     /**
47      * 상태 변경 함수입니다.
48      * 상태를 제거할 수도 있고, 상태를 추가할 수도 있습니다.
49      */
50     /**
51      * 만일 현재 추가상태라면?
52      */
53     if (this->_isAdding)
54     {
55         /**
56          * 만일 스택에 상태들이 쌓여있다면?
57          */
58         if (!this->_states.empty())
59         {
60             /**
61               * 교체 상태가 교체 상태라면?
62               */
63             if (this->_isReplacing)
64             {
65                 /**
66                   * 교체 상태이니 스택의 맨 위의 요소를 제거합니다.
67                   */
68                 this->_states.pop();
69             }
70             /**
71               * 만일 상태가 교체 상태가 아니라면?
72               */
73             else
74             {
75                 /**
76                   * 교체 상태가 아니니 스택의 맨 위의 상태를 잠시 멈춥니다.
77                   */
78                 this->_states.top()->Pause();
79             }
80         }
81         /**
82           * 현재 추가 상태입니다. 현재 State를 소유권을 이전해서 스택에 쌓아 놓습니다.
83           */
84         this->_states.push(move(this->_newState));
85         /**
86           * 성공 종료합니다. 스택의 맨 위의 상태를 초기화(다음 상황을 대비해서 정소 해 줍니다. )합니다.
87           */
88         this->_states.top()->Init();
89         /**
90           * 더 이상 추가 상태가 아닙니다.
91           */
92         this->_isAdding = false;
93     }
94 }

```

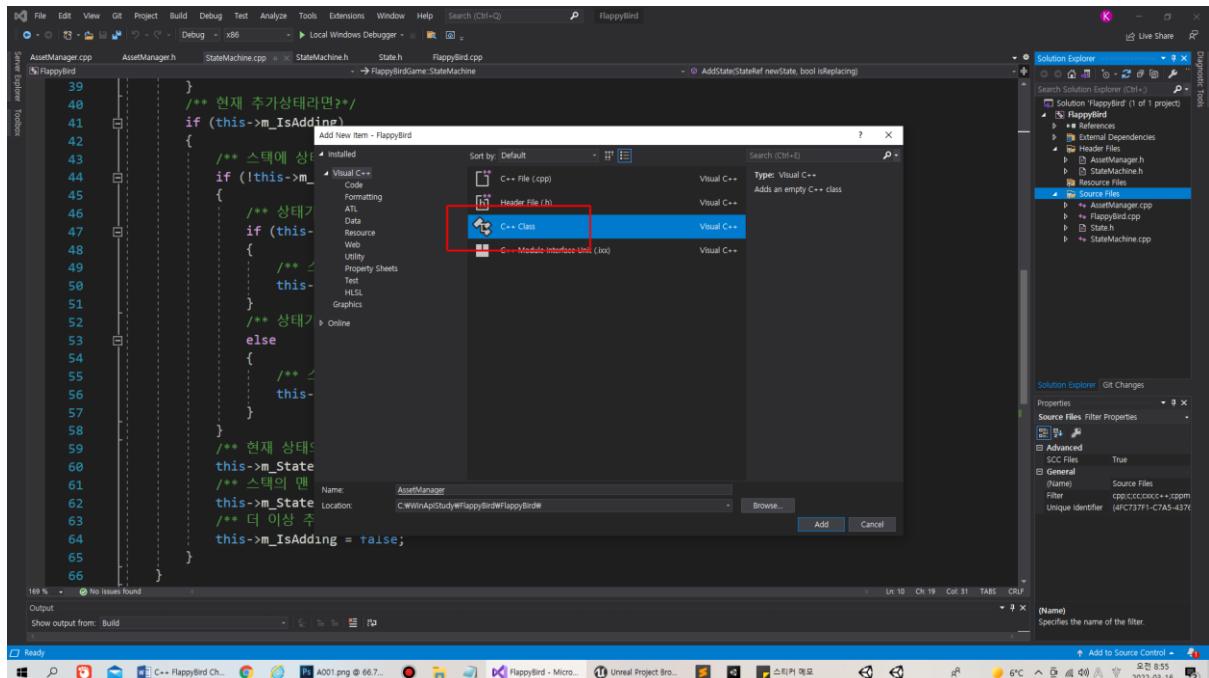
게임에 필요한 에셋들을 로드해야 합니다. AssetManager라는 이름으로 클래스를 만들어 주도록 합니다.



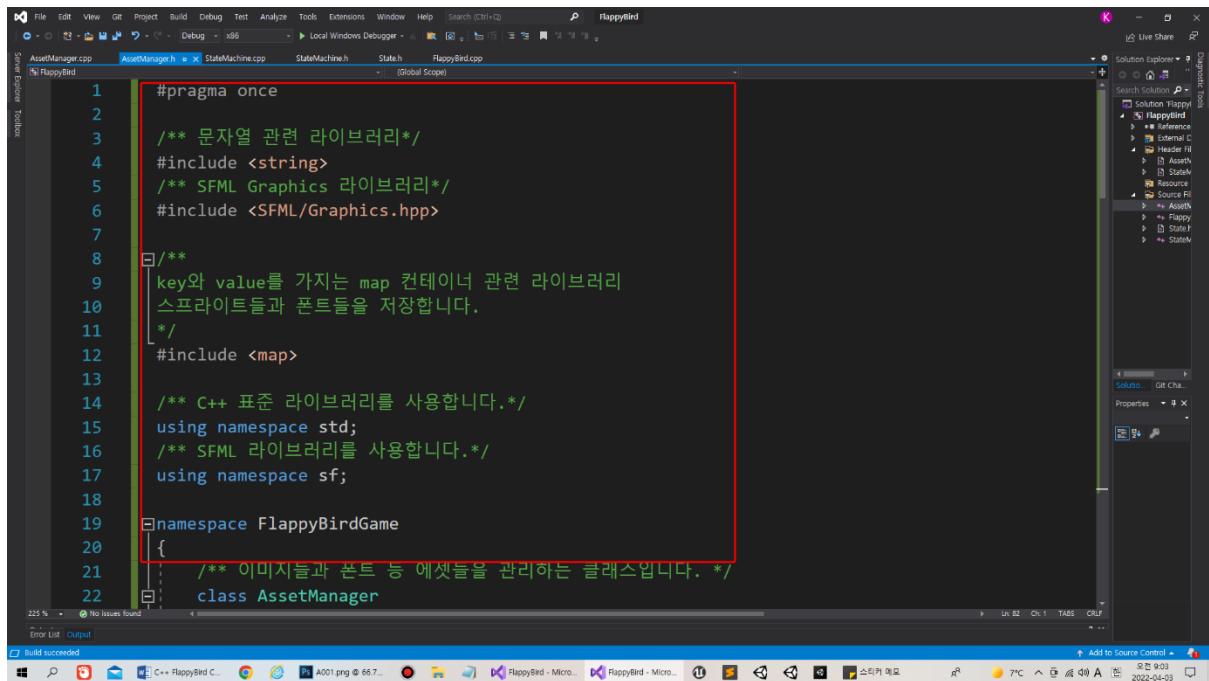
```

39     }
40     /**
41       * 현재 추가상태라면?
42       */
43     if (this->_m_IsAdding)
44     {
45         /**
46           * 스택에 상태들이 쌓여 있다면?
47           */
48         if (!this->_m_States.empty())
49         {
50             /**
51               * 상태가 교체 상태라면?
52               */
53             if (this->_m_IsReplacing)
54             {
55                 /**
56                   * 스택의 맨 위의 상태를 제거 합니다.
57                   */
58                 this->_m_States.pop();
59             }
60             /**
61               * 상태가 교체 상태가 아니라면?
62               */
63             else
64             {
65                 /**
66                   * 스택에 쌓여있는 맨 위의 상태를 잠시 멈춥니다.
67                   */
68                 this->_m_States.top()->Pause();
69             }
70         }
71         /**
72           * 현재 상태의 소유권을 이전해서 스택에 쌓아 놓습니다.
73           */
74         this->_m_States.push(move(this->_m_NewState));
75         /**
76           * 스택의 맨 위의 상태를 초기화 합니다.
77           */
78         this->_m_States.top()->Init();
79         /**
80           * 더 이상 추가 상태가 아닙니다.
81           */
82         this->_m_IsAdding = false;
83     }
84 }

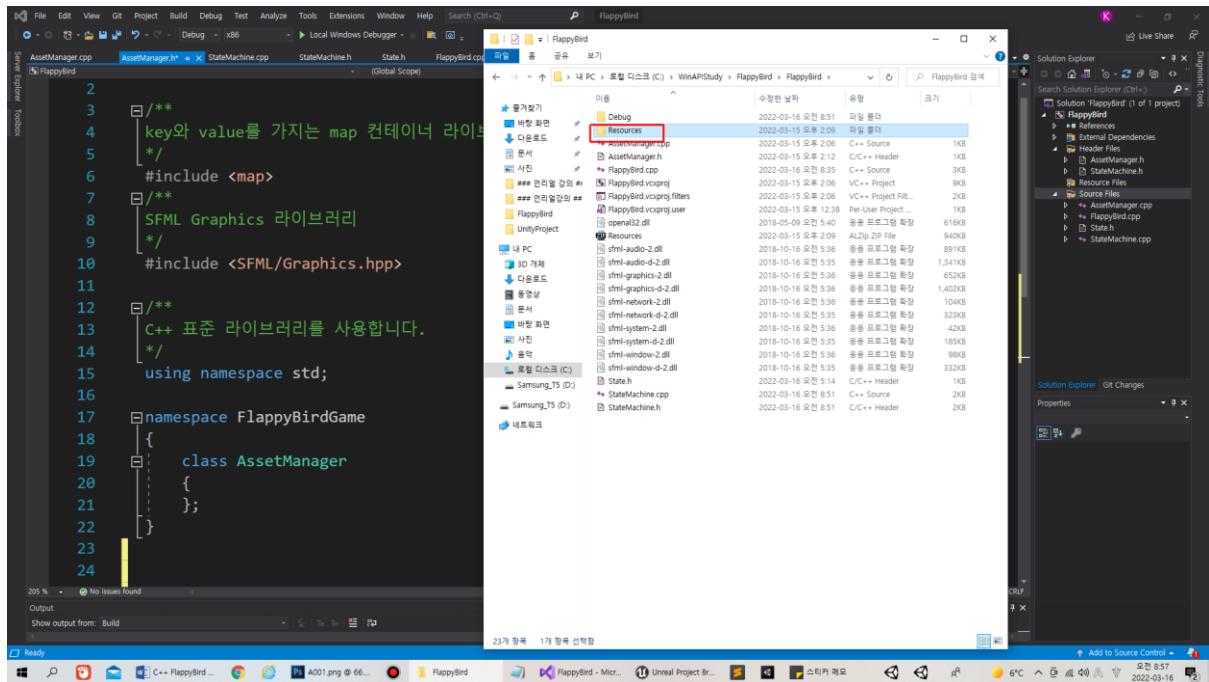
```



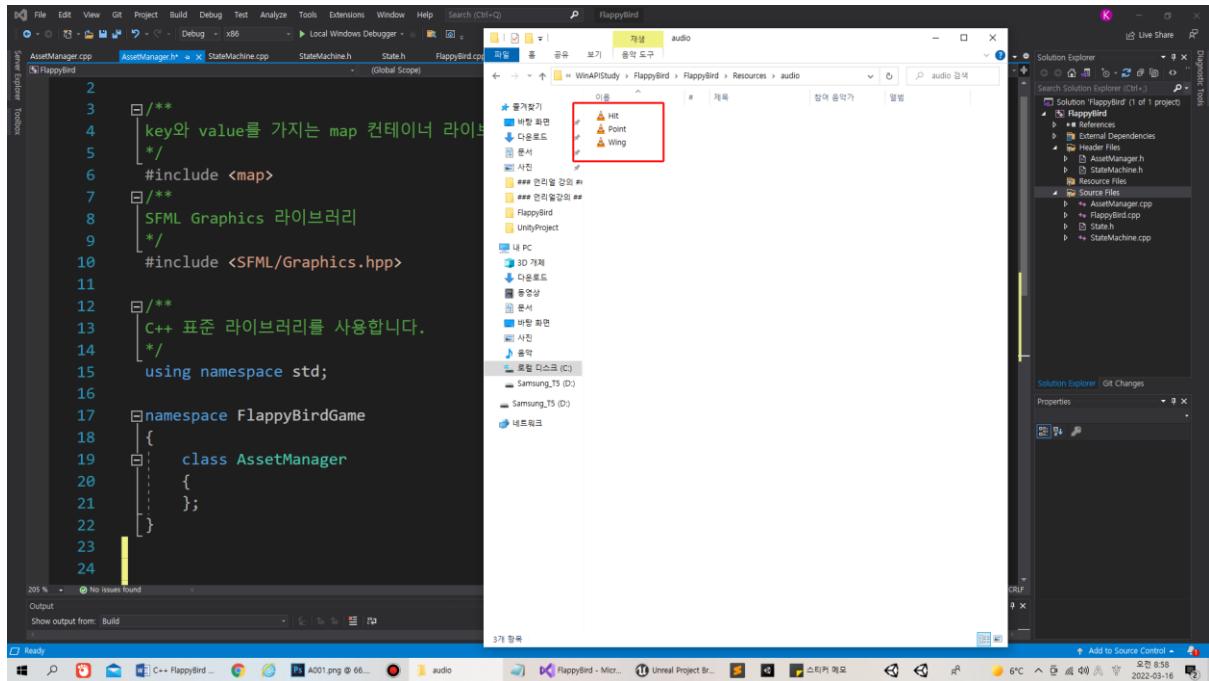
헤더 파일에서 필요한 라이브러리를 추가해 주도록 합니다.



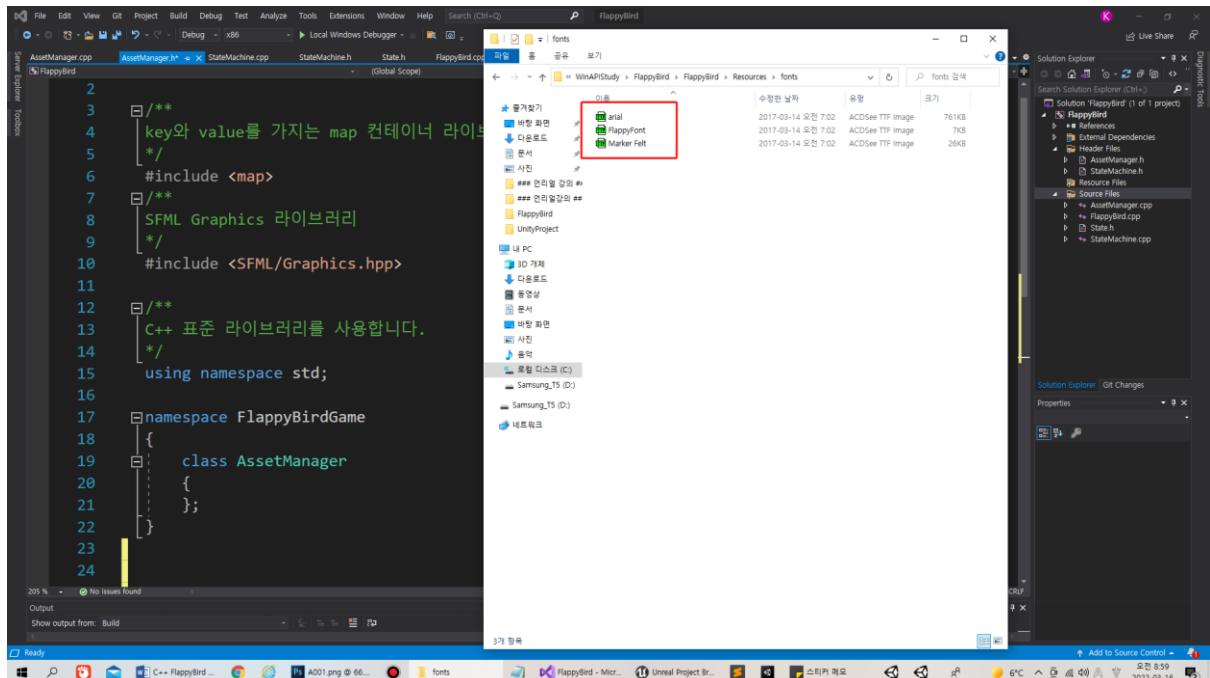
Resources 폴더를 만들어 주도록 합니다.



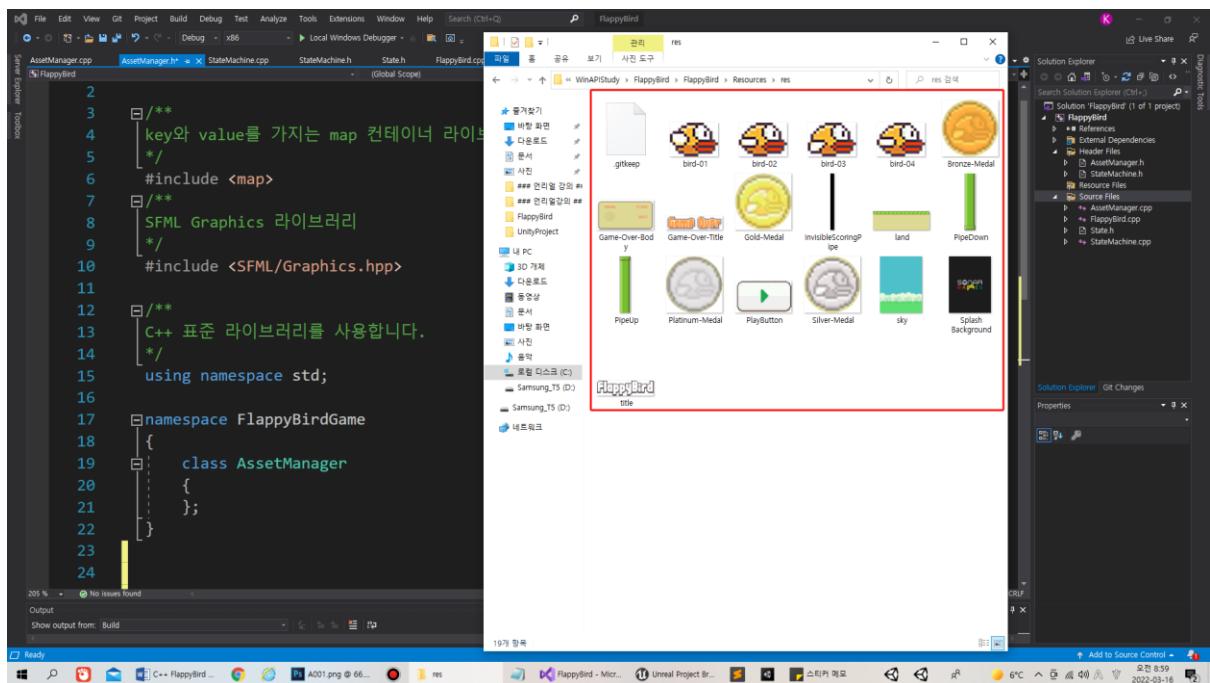
사운드들이 보입니다.



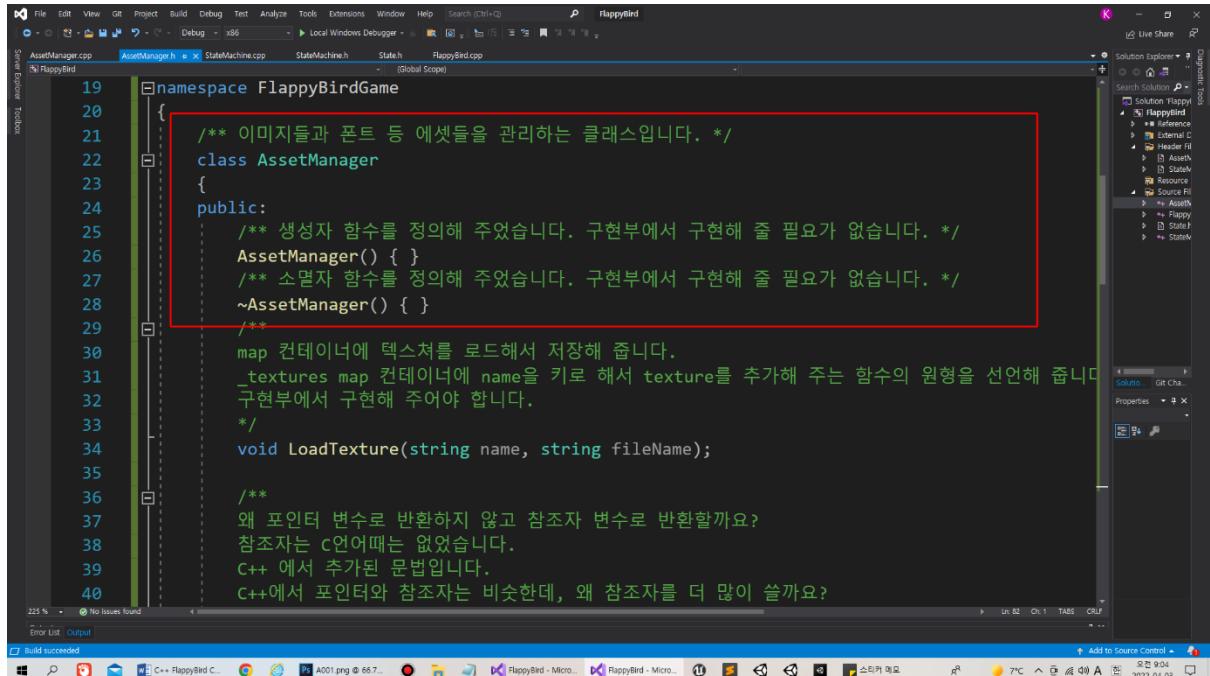
폰트들이 보입니다.



이미지들이 보입니다.

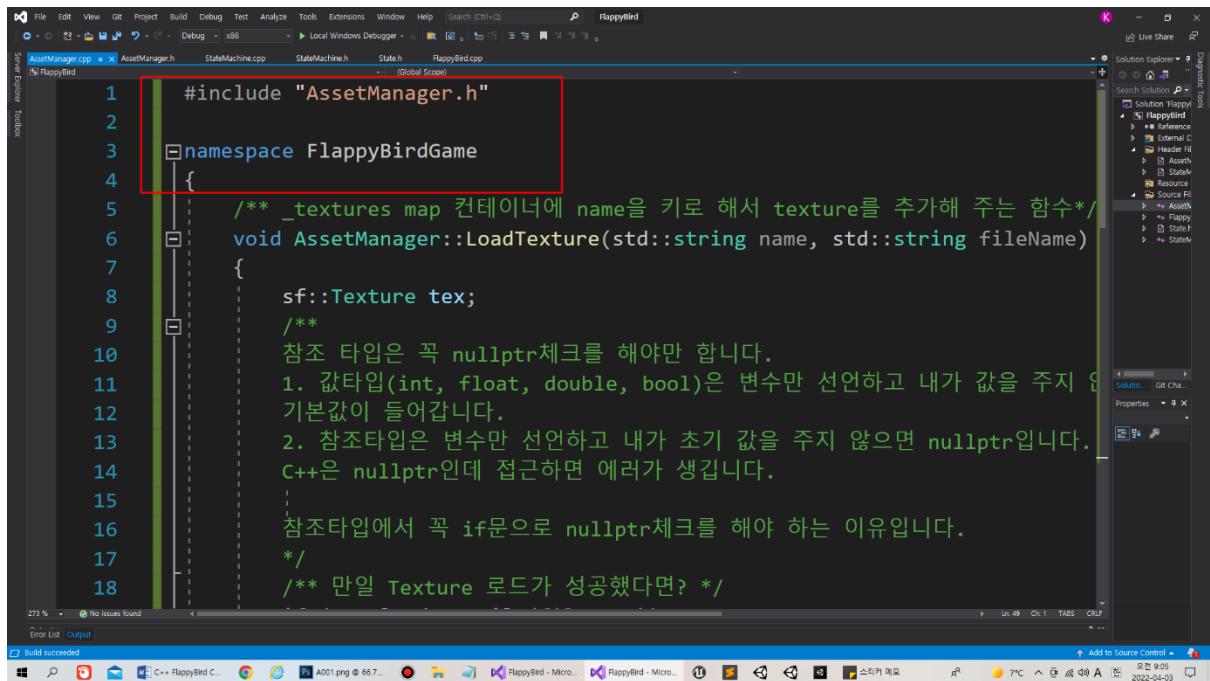


AssetManager 헤더 파일로 가서 생성자 함수와 소멸자 함수를 만들어 주도록 합니다.



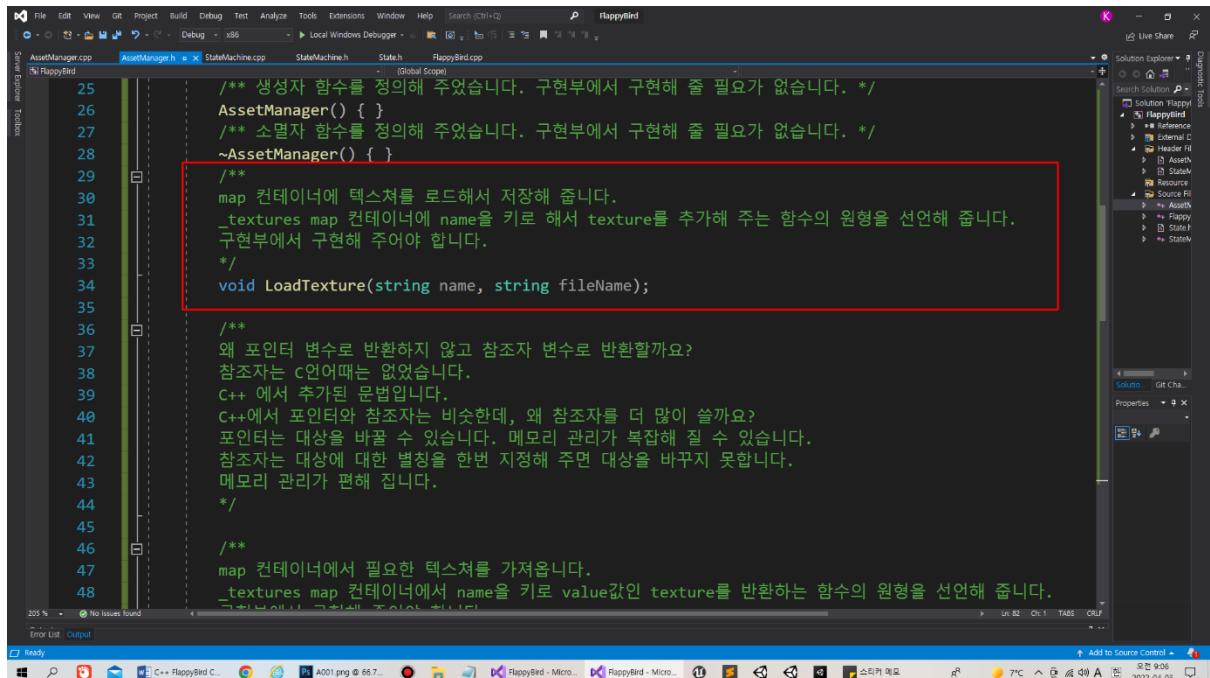
```
19  namespace FlappyBirdGame
20  {
21      /**
22      * 이미지들과 폰트 등 에셋들을 관리하는 클래스입니다.
23      */
24      class AssetManager
25      {
26          /**
27          * 생성자 함수를 정의해 주었습니다. 구현부에서 구현해 줄 필요가 없습니다.
28          */
29          AssetManager() { }
30          /**
31          * 소멸자 함수를 정의해 주었습니다. 구현부에서 구현해 줄 필요가 없습니다.
32          */
33          ~AssetManager() { }
34          /**
35          * map 컨테이너에 텍스처를 로드해서 저장해 줍니다.
36          */
37          void LoadTexture(string name, string fileName);
38
39          /**
40          * 왜 포인터 변수로 반환하지 않고 참조자 변수로 반환할까요?
41          * 참조자는 C언어때는 없었습니다.
42          * C++에서 추가된 문법입니다.
43          * C++에서 포인터와 참조자는 비슷한데, 왜 참조자를 더 많이 쓸까요?
44      }
```

구현부로 가서 필요한 라이브러리를 등록해 주도록 합니다.



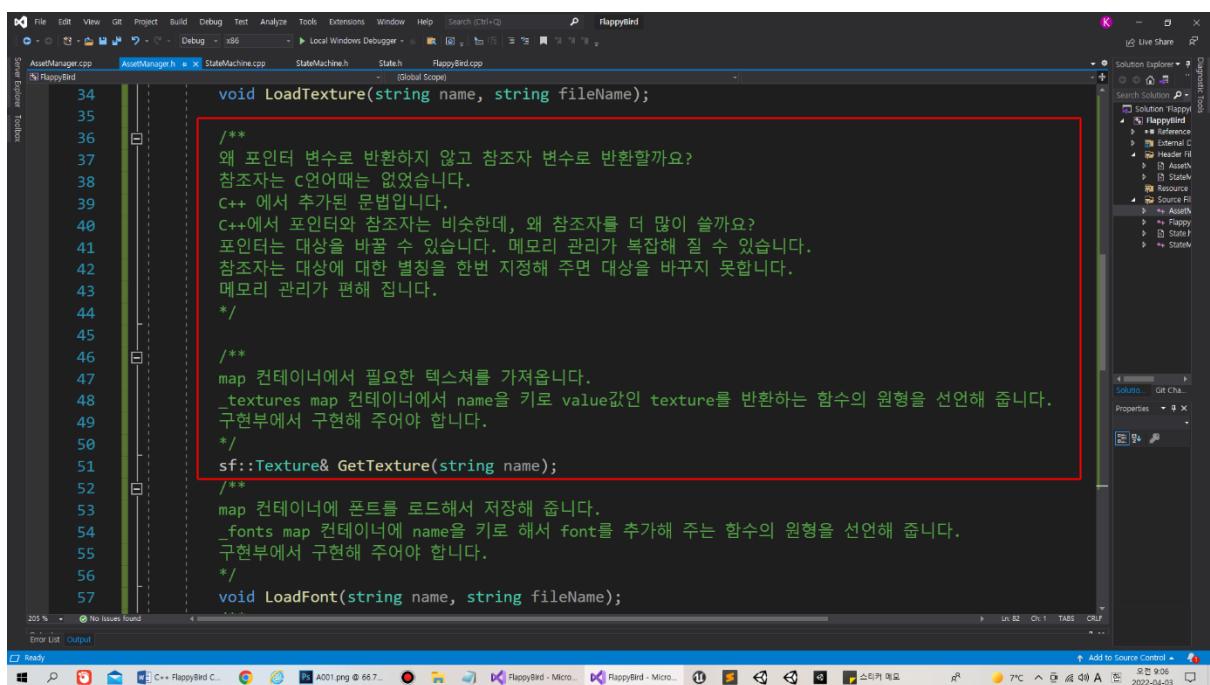
```
1 #include "AssetManager.h"
2
3 namespace FlappyBirdGame
4 {
5     /**
6      * _textures map 컨테이너에 name을 키로 해서 texture를 추가해 주는 함수*/
7     void AssetManager::LoadTexture(std::string name, std::string fileName)
8     {
9         sf::Texture tex;
10
11         /**
12         * 참조 타입은 꼭 nullptr체크를 해야만 합니다.
13         * 1. 값타입(int, float, double, bool)은 변수만 선언하고 내가 값을 주지 않으면 기본값이 들어갑니다.
14         * 2. 참조타입은 변수만 선언하고 내가 초기 값을 주지 않으면 nullptr입니다.
15         * C++은 nullptr인데 접근하면 에러가 생깁니다.
16
17         /**
18         * 참조타입에서 꼭 if문으로 nullptr체크를 해야 하는 이유입니다.
19
20         /**
21         * 만일 Texture 로드가 성공했다면? */
22     }
```

헤더 파일로 가서 필요한 함수들과 변수들을 선언해 줍니다.



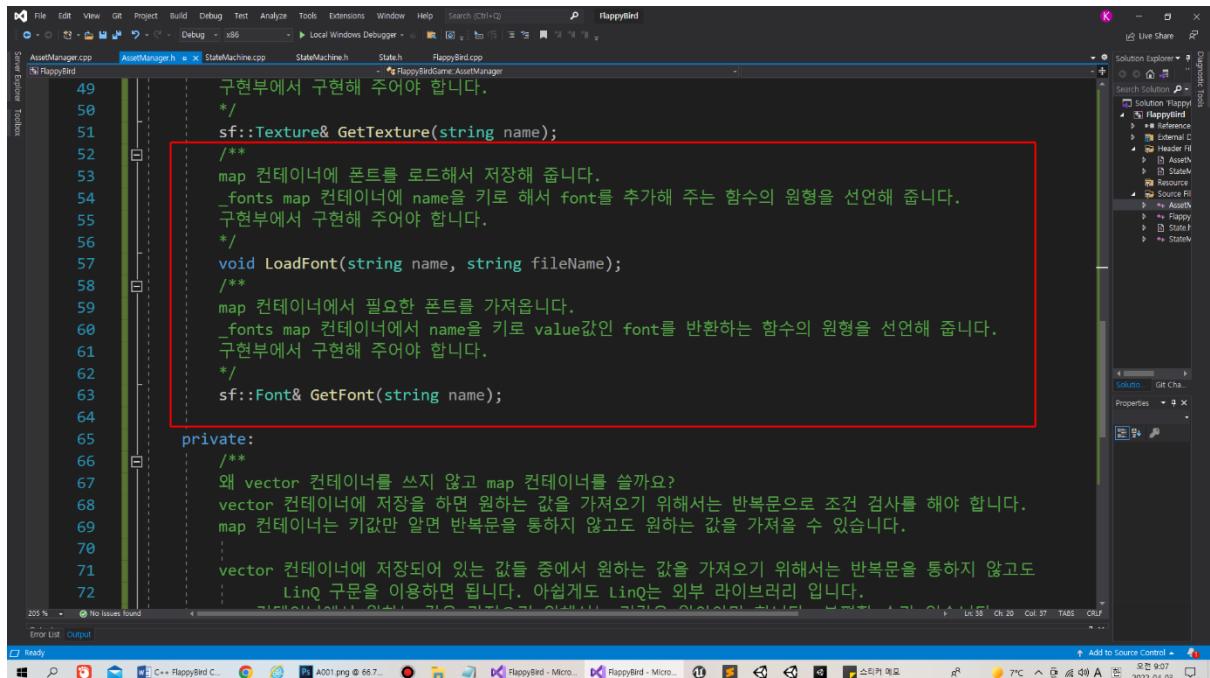
```
25 	/** 생성자 함수를 정의해 주었습니다. 구현부에서 구현해 줄 필요가 없습니다. */
26 	AssetManager() { }
27 	/** 소멸자 함수를 정의해 주었습니다. 구현부에서 구현해 줄 필요가 없습니다. */
28 	~AssetManager() { }
29
30 	/**
31 	map 컨테이너에 텍스쳐를 로드해서 저장해 줍니다.
32 	_textures map 컨테이너에 name을 키로 해서 texture를 추가해 주는 함수의 원형을 선언해 줍니다.
33 	구현부에서 구현해 주어야 합니다.
34 	*/
35 	void LoadTexture(string name, string fileName);
36
37 	/**
38 	왜 포인터 변수로 반환하지 않고 참조자 변수로 반환할까요?
39 	참조자는 C언어에는 없었습니다.
40 	C++에서 추가된 문법입니다.
41 	C++에서 포인터와 참조자는 비슷한데, 왜 참조자를 더 많이 쓸까요?
42 	포인터는 대상을 바꿀 수 있습니다. 메모리 관리가 복잡해 질 수 있습니다.
43 	참조자는 대상에 대한 별칭을 한번 지정해 주면 대상을 바꾸지 못합니다.
44 	메모리 관리가 편해집니다.
45
46 	/**
47 	map 컨테이너에서 필요한 텍스쳐를 가져옵니다.
48 	_textures map 컨테이너에서 name을 키로 value값인 texture를 반환하는 함수의 원형을 선언해 줍니다.
49
50
51
52
53
54
55
56
57
```

2022-04-03 09:00 7°C A

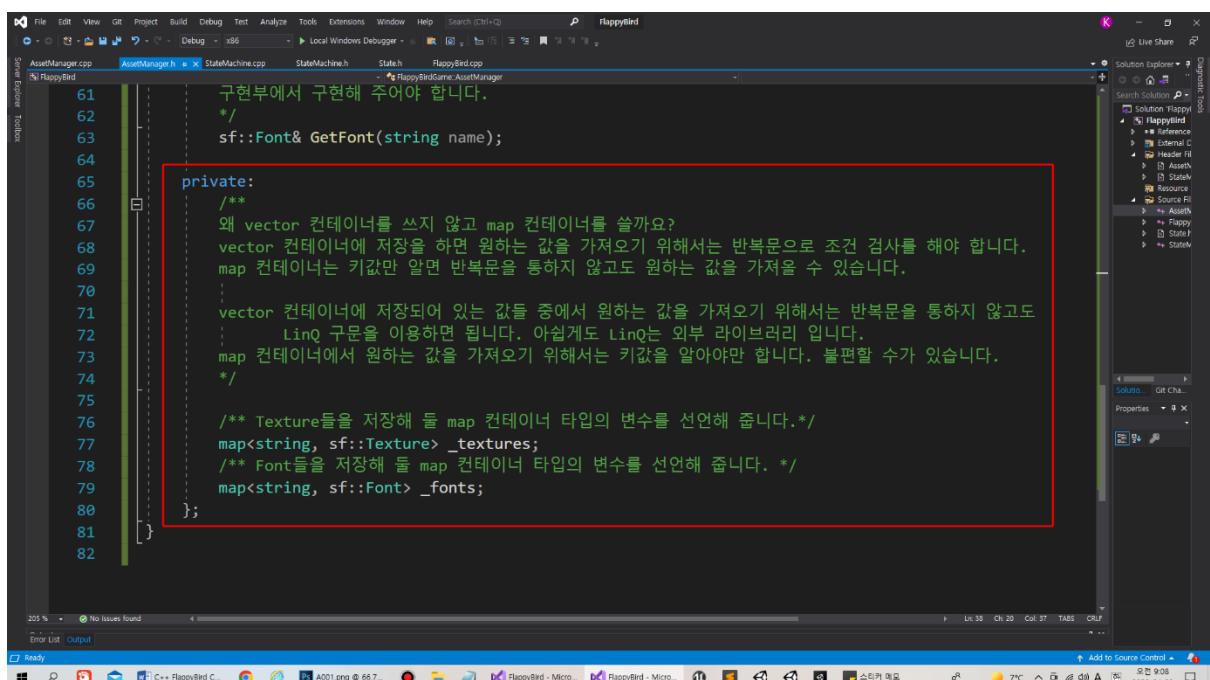


```
34 	void LoadTexture(string name, string fileName);
35
36 	/**
37 	왜 포인터 변수로 반환하지 않고 참조자 변수로 반환할까요?
38 	참조자는 C언어에는 없었습니다.
39 	C++에서 추가된 문법입니다.
40 	C++에서 포인터와 참조자는 비슷한데, 왜 참조자를 더 많이 쓸까요?
41 	포인터는 대상을 바꿀 수 있습니다. 메모리 관리가 복잡해 질 수 있습니다.
42 	참조자는 대상에 대한 별칭을 한번 지정해 주면 대상을 바꾸지 못합니다.
43 	메모리 관리가 편해집니다.
44
45 	/**
46 	map 컨테이너에서 필요한 텍스쳐를 가져옵니다.
47 	_textures map 컨테이너에서 name을 키로 value값인 texture를 반환하는 함수의 원형을 선언해 줍니다.
48 	구현부에서 구현해 주어야 합니다.
49
50
51
52
53
54
55
56
57
```

2022-04-03 09:06 7°C A

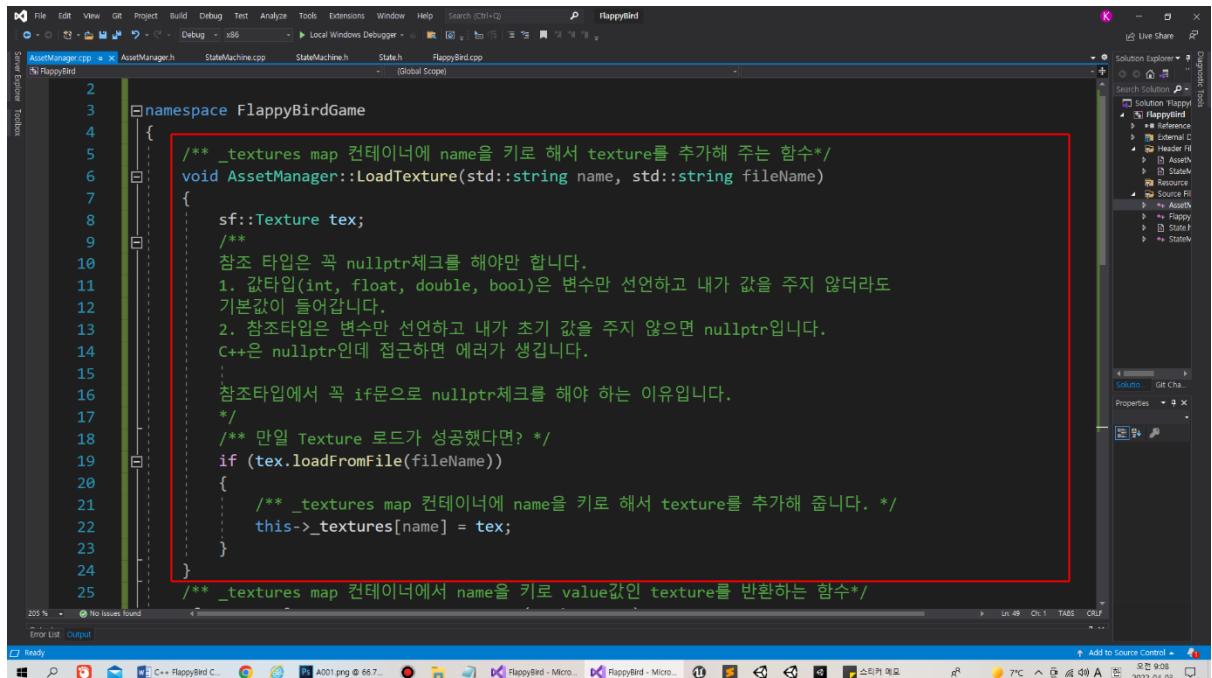


```
49     구현부에서 구현해 주어야 합니다.
50
51     */
52     sf::Texture& GetTexture(string name);
53
54     /**
55      map 컨테이너에 폰트를 로드해서 저장해 줍니다.
56      _fonts map 컨테이너에 name을 키로 해서 font를 추가해 주는 함수의 원형을 선언해 줍니다.
57      구현부에서 구현해 주어야 합니다.
58
59     void LoadFont(string name, string fileName);
60
61     /**
62      map 컨테이너에서 필요한 폰트를 가져옵니다.
63      _fonts map 컨테이너에서 name을 키로 value값인 font를 반환하는 함수의 원형을 선언해 줍니다.
64      구현부에서 구현해 주어야 합니다.
65
66     sf::Font& GetFont(string name);
67
68     /**
69      왜 vector 컨테이너를 쓰지 않고 map 컨테이너를 쓸까요?
70      vector 컨테이너에 저장을 하면 원하는 값을 가져오기 위해서는 반복문으로 조건 검사를 해야 합니다.
71      map 컨테이너는 키값만 알면 반복문을 통하지 않고도 원하는 값을 가져올 수 있습니다.
72
73      vector 컨테이너에 저장되어 있는 값을 중에서 원하는 값을 가져오기 위해서는 반복문을 통하지 않고도
74      LinQ 구문을 이용하면 됩니다. 아쉽게도 LinQ는 외부 라이브러리 입니다.
```

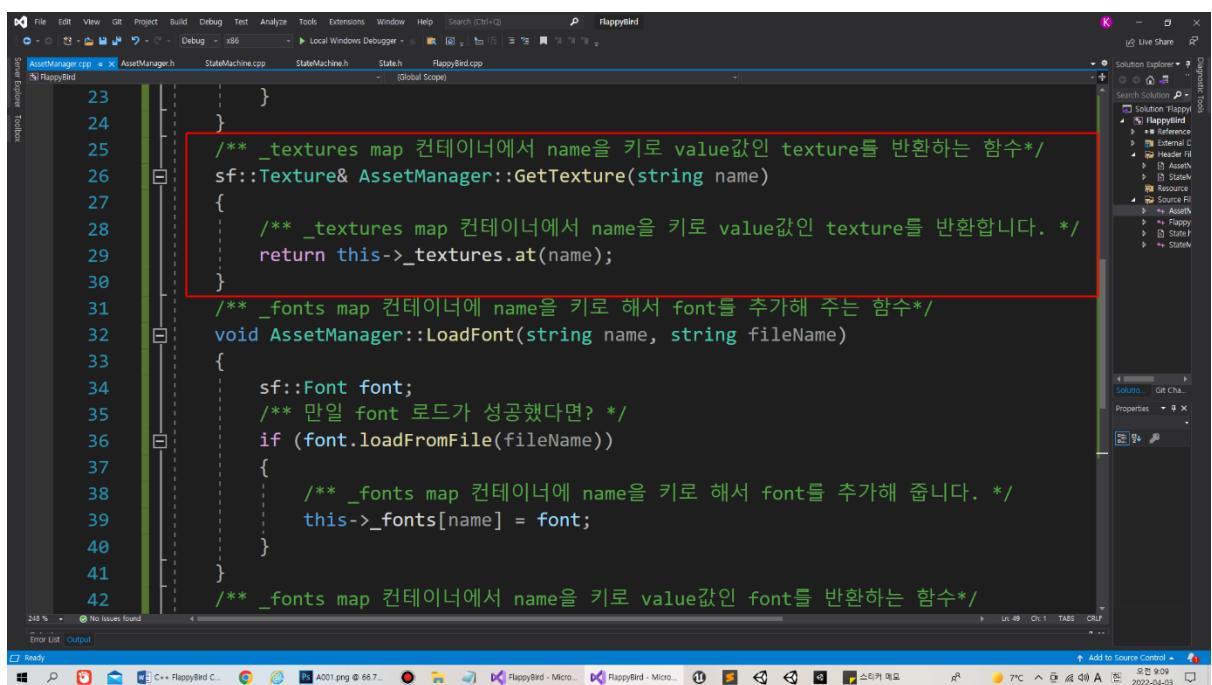


```
61     구현부에서 구현해 주어야 합니다.
62
63     sf::Font& GetFont(string name);
64
65     /**
66      왜 vector 컨테이너를 쓰지 않고 map 컨테이너를 쓸까요?
67      vector 컨테이너에 저장을 하면 원하는 값을 가져오기 위해서는 반복문으로 조건 검사를 해야 합니다.
68      map 컨테이너는 키값만 알면 반복문을 통하지 않고도 원하는 값을 가져올 수 있습니다.
69
70      vector 컨테이너에 저장되어 있는 값을 중에서 원하는 값을 가져오기 위해서는 반복문을 통하지 않고도
71      LinQ 구문을 이용하면 됩니다. 아쉽게도 LinQ는 외부 라이브러리 입니다.
72      map 컨테이너에서 원하는 값을 가져오기 위해서는 키값을 알아야만 합니다. 불편할 수가 있습니다.
73
74
75
76     /**
77      Texture들을 저장해 둘 map 컨테이너 타입의 변수를 선언해 줍니다.*/
78     map<string, sf::Texture> _textures;
79
80     /**
81      Font들을 저장해 둘 map 컨테이너 타입의 변수를 선언해 줍니다. */
82     map<string, sf::Font> _fonts;
83 }
```

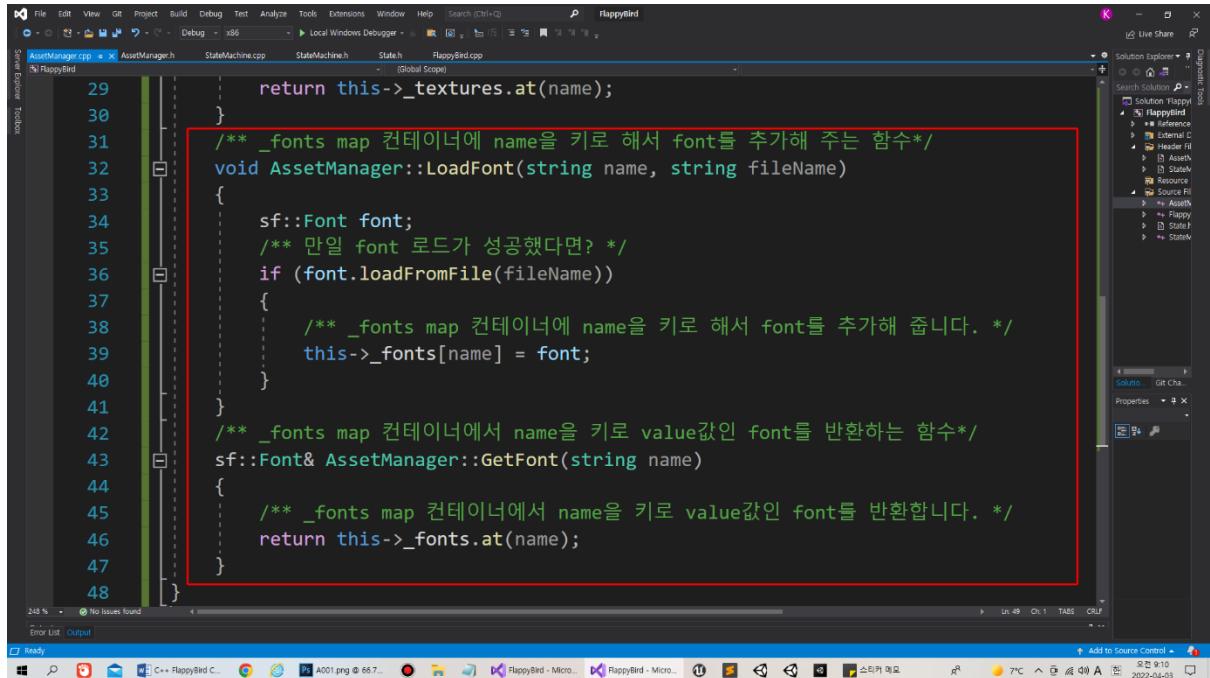
구현부에서 구현해 주도록 합니다.



```
2
3     namespace FlappyBirdGame
4     {
5         /** _textures map 컨테이너에 name을 키로 해서 texture를 추가해 주는 함수*/
6         void AssetManager::LoadTexture(std::string name, std::string fileName)
7         {
8             sf::Texture tex;
9             /**
10             참조 타입은 꼭 nullptr체크를 해야만 합니다.
11             1. 값타입(int, float, double, bool)은 변수만 선언하고 내가 값을 주지 않더라도
12             기본값이 들어갑니다.
13             2. 참조타입은 변수만 선언하고 내가 초기 값을 주지 않으면 nullptr입니다.
14             C++은 nullptr인데 접근하면 에러가 생깁니다.
15
16             참조타입에서 꼭 if문으로 nullptr체크를 해야 하는 이유입니다.
17
18             /**
19             만일 Texture 로드가 성공했다면? */
20             if (tex.loadFromFile(fileName))
21             {
22                 /** _textures map 컨테이너에 name을 키로 해서 texture를 추가해 줍니다. */
23                 this->_textures[name] = tex;
24             }
25
26         /** _textures map 컨테이너에서 name을 키로 value값인 texture를 반환하는 함수*/
27     }
```

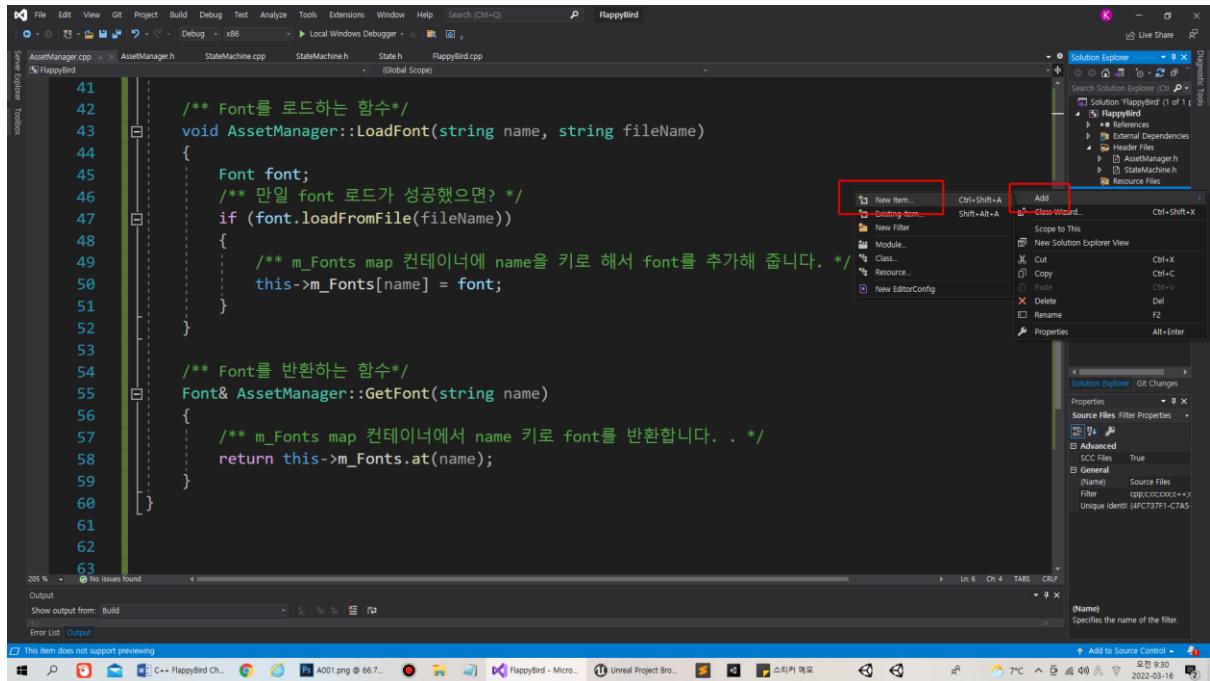


```
23
24
25     /**
26     sf::Texture& AssetManager::GetTexture(string name)
27     {
28         /** _textures map 컨테이너에서 name을 키로 value값인 texture를 반환합니다. */
29         return this->_textures.at(name);
30     }
31
32     /** _fonts map 컨테이너에 name을 키로 해서 font를 추가해 주는 함수*/
33     void AssetManager::LoadFont(string name, string fileName)
34     {
35         sf::Font font;
36         /**
37         만일 font 로드가 성공했다면? */
38         if (font.loadFromFile(fileName))
39         {
40             /** _fonts map 컨테이너에 name을 키로 해서 font를 추가해 줍니다. */
41             this->_fonts[name] = font;
42
43         /** _fonts map 컨테이너에서 name을 키로 value값인 font를 반환하는 함수*/
44     }
```

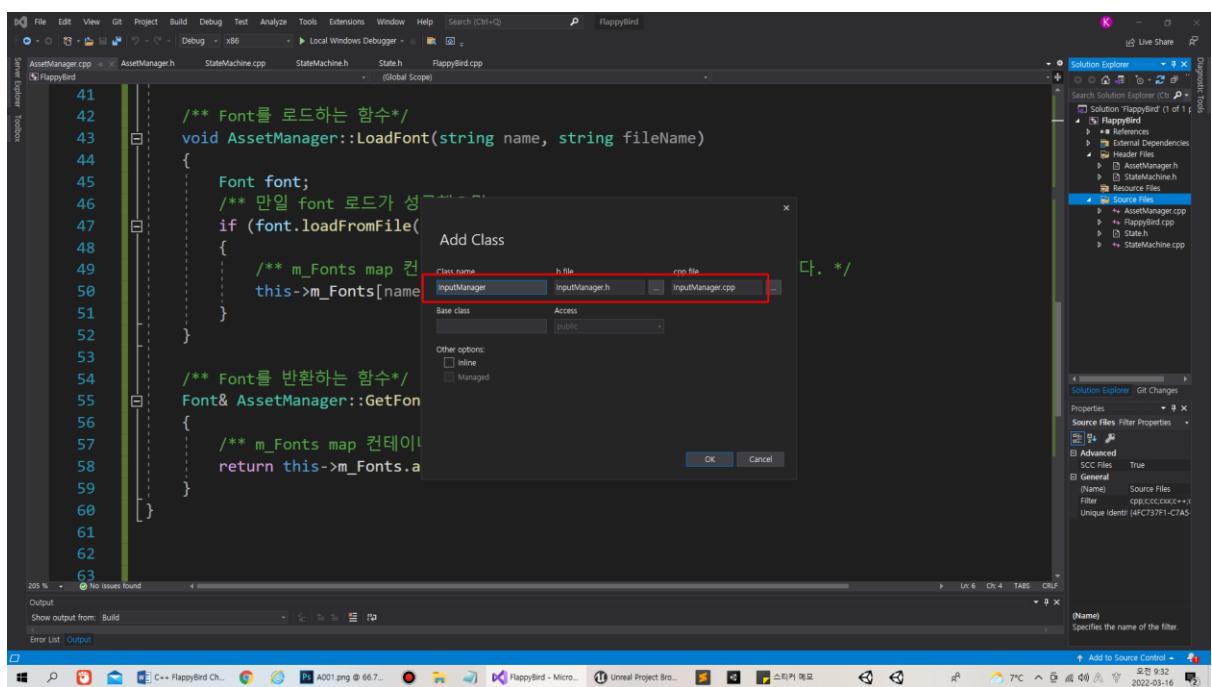
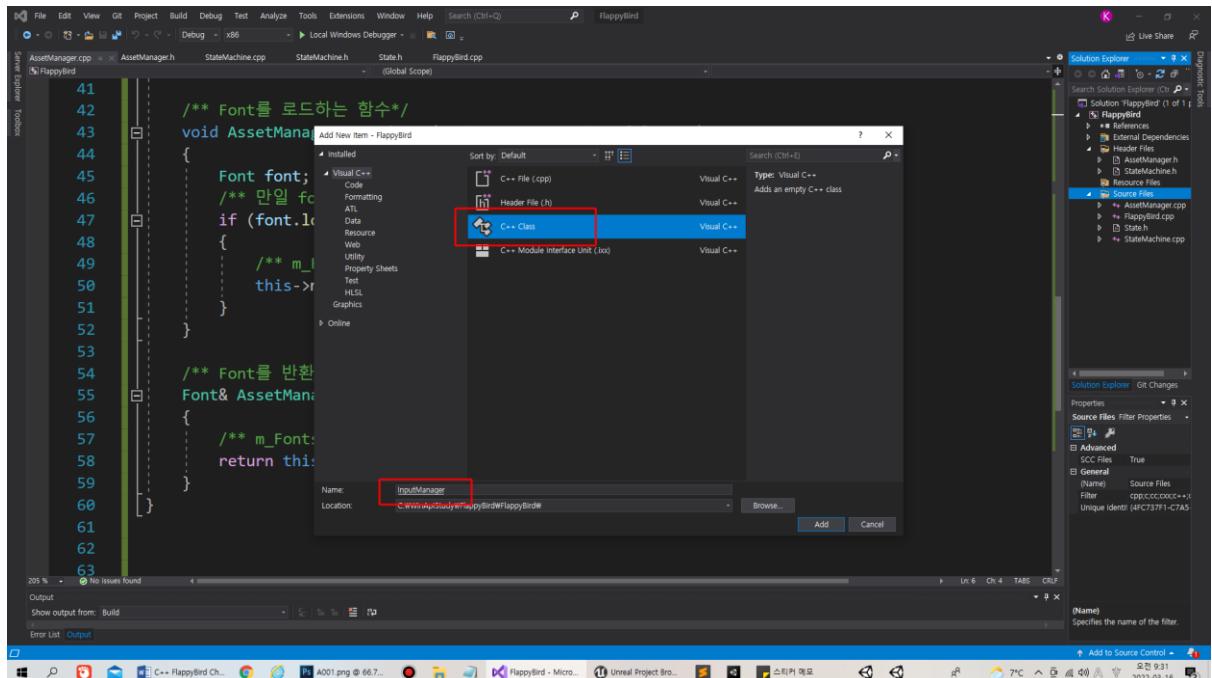


```
29         return this->textures.at(name);
30     }
31
32     /** _fonts map 컨테이너에 name을 키로 해서 font를 추가해 주는 함수*/
33     void AssetManager::LoadFont(string name, string fileName)
34     {
35         sf::Font font;
36         /** 만일 font 로드가 성공했다면? */
37         if (font.loadFromFile(fileName))
38         {
39             /** _fonts map 컨테이너에 name을 키로 해서 font를 추가해 줍니다. */
40             this->fonts[name] = font;
41         }
42
43         /** _fonts map 컨테이너에서 name을 키로 value값인 font를 반환하는 함수*/
44         sf::Font& AssetManager::GetFont(string name)
45         {
46             /** _fonts map 컨테이너에서 name을 키로 value값인 font를 반환합니다. */
47             return this->fonts.at(name);
48         }
49     }
```

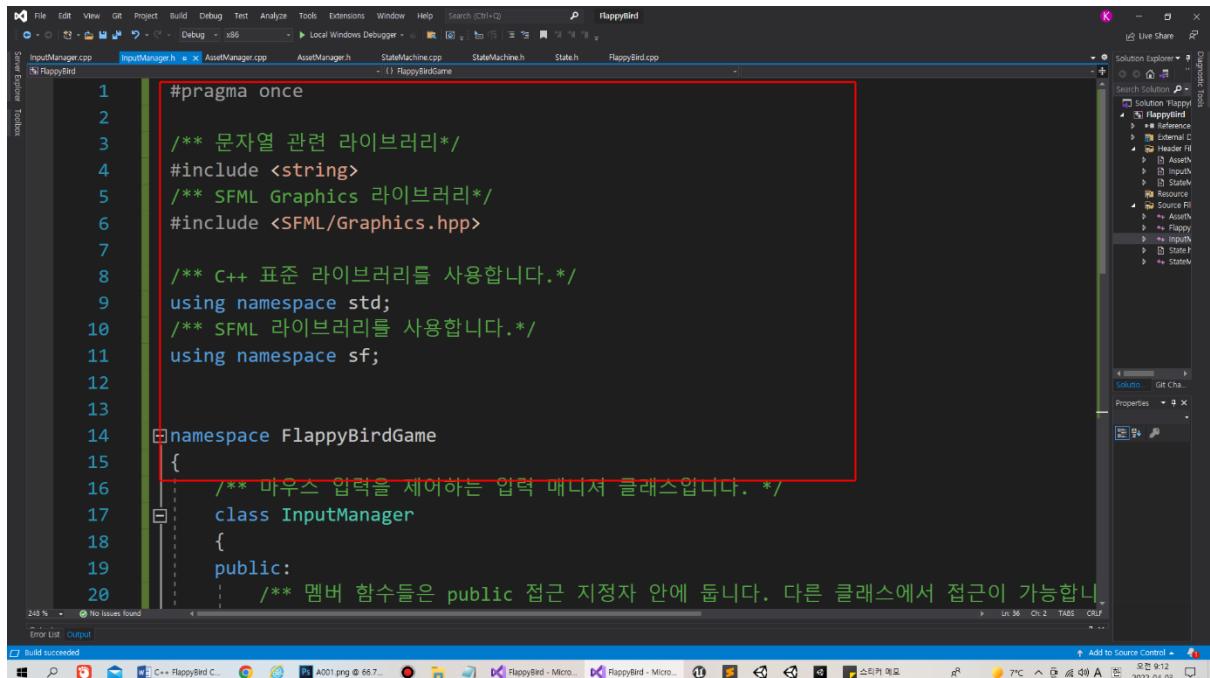
이제 인풋 관련을 담당하는 InputManager클래스를 만들어 주도록 합니다.



```
41
42
43     /** Font를 로드하는 함수*/
44     void AssetManager::LoadFont(string name, string fileName)
45     {
46         Font font;
47         /** 만일 font 로드가 성공했으면? */
48         if (font.loadFromFile(fileName))
49         {
50             /** m_Fonts map 컨테이너에 name을 키로 해서 font를 추가해 줍니다. */
51             this->m_Fonts[name] = font;
52         }
53
54         /** Font를 반환하는 함수*/
55         Font& AssetManager::GetFont(string name)
56         {
57             /** m_Fonts map 컨테이너에서 name 키로 font를 반환합니다. . */
58             return this->m_Fonts.at(name);
59         }
60     }
61
62
63 }
```

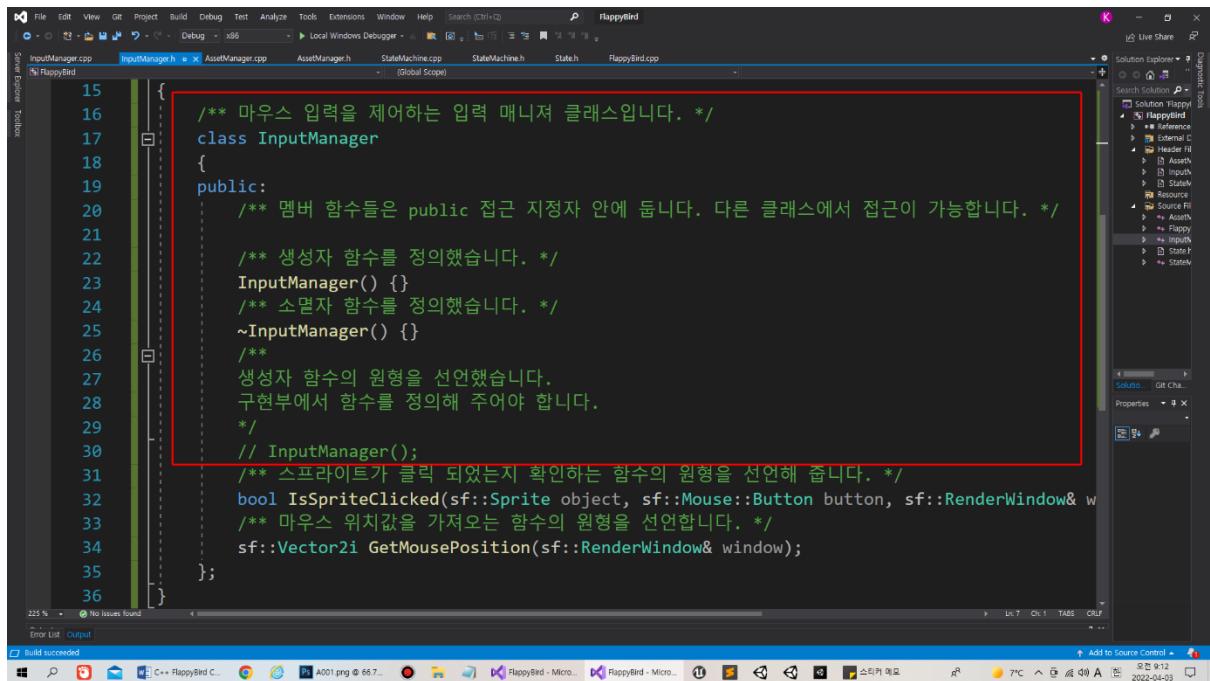


헤더 파일로 가서 필요한 라이브러리를 추가해 주도록 합니다.



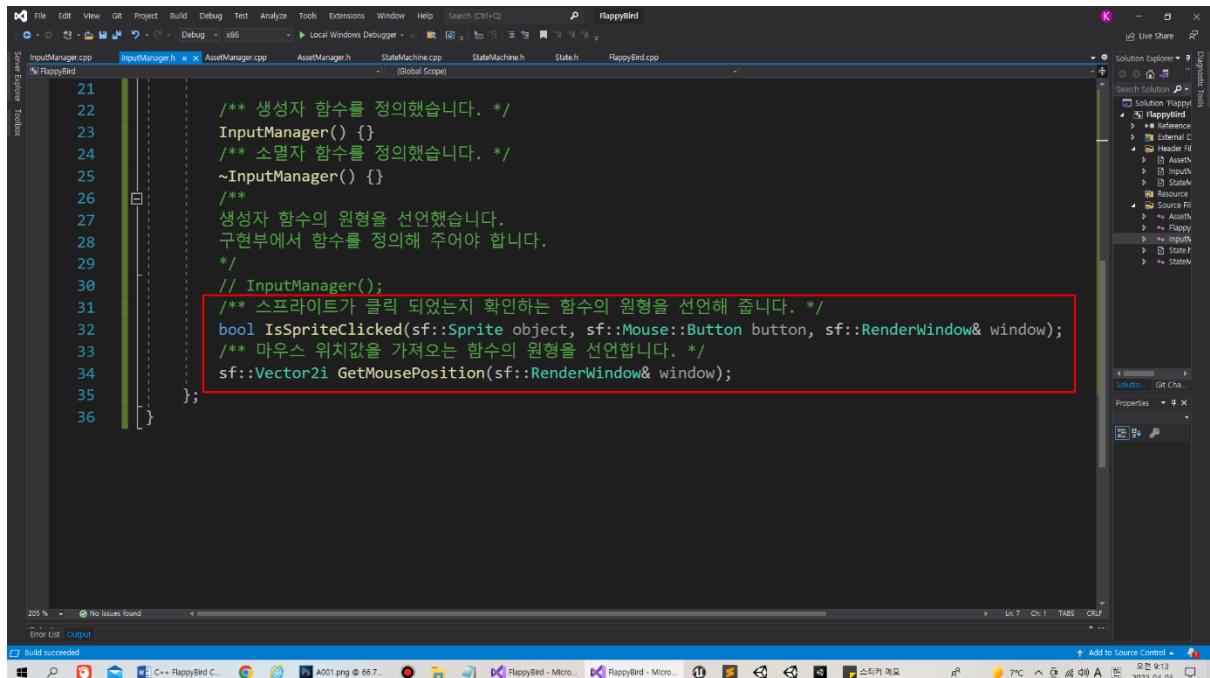
```
1 #pragma once
2
3 /** 문자열 관련 라이브러리*/
4 #include <string>
5 /** SFML Graphics 라이브러리*/
6 #include <SFML/Graphics.hpp>
7
8 /** C++ 표준 라이브러리를 사용합니다.*/
9 using namespace std;
10 /** SFML 라이브러리를 사용합니다.*/
11 using namespace sf;
12
13
14 namespace FlappyBirdGame
15 {
16     /** 마우스 입력을 처리하는 입력 매니저 클래스입니다. */
17     class InputManager
18     {
19     public:
20         /** 멤버 함수들은 public 접근 자정자 안에 둡니다. 다른 클래스에서 접근이 가능합니다. */
21
22         /**
23          * 생성자 함수를 정의했습니다.
24          */
25         InputManager() {}
26         /**
27          * 소멸자 함수를 정의했습니다.
28          */
29         ~InputManager() {}
30
31         /**
32          * 스프라이트가 클릭되었는지 확인하는 함수의 원형을 선언해 줍니다.
33          */
34         bool IsSpriteClicked(sf::Sprite object, sf::Mouse::Button button, sf::RenderWindow& window);
35
36     };
37 }
```

생성자, 소멸자 함수를 만들어 주도록 합니다.



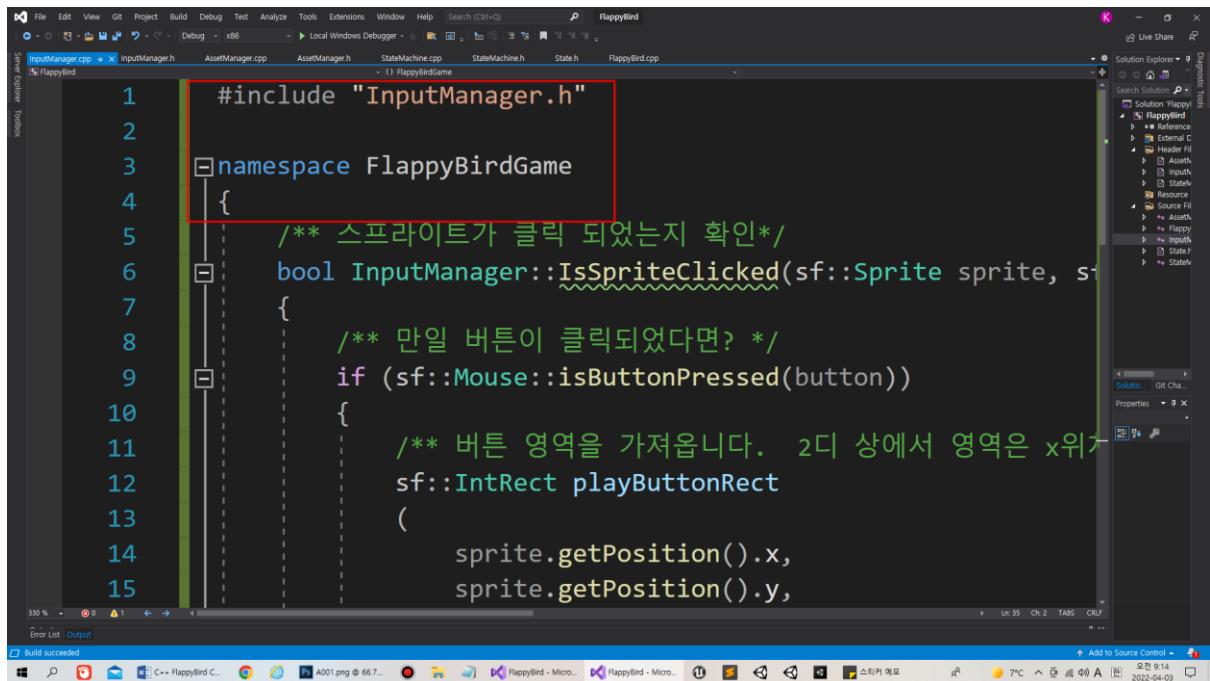
```
15 /**
16  * 마우스 입력을 처리하는 입력 매니저 클래스입니다.
17  */
18 class InputManager
19 {
20 public:
21     /**
22      * 생성자 함수를 정의했습니다.
23      */
24     InputManager() {}
25     /**
26      * 소멸자 함수를 정의했습니다.
27      */
28     ~InputManager() {}
29
30     /**
31      * 스프라이트가 클릭되었는지 확인하는 함수의 원형을 선언해 줍니다.
32      */
33     bool IsSpriteClicked(sf::Sprite object, sf::Mouse::Button button, sf::RenderWindow& window);
34
35     /**
36      * 마우스 위치값을 가져오는 함수의 원형을 선언합니다.
37      */
38     sf::Vector2i GetMousePosition(sf::RenderWindow& window);
39
40 };
41 }
```

스프라이트가 클릭되었는지 확인하는 함수와 마우스 위치값을 가져오는 함수의 원형을 선언해 줍니다.



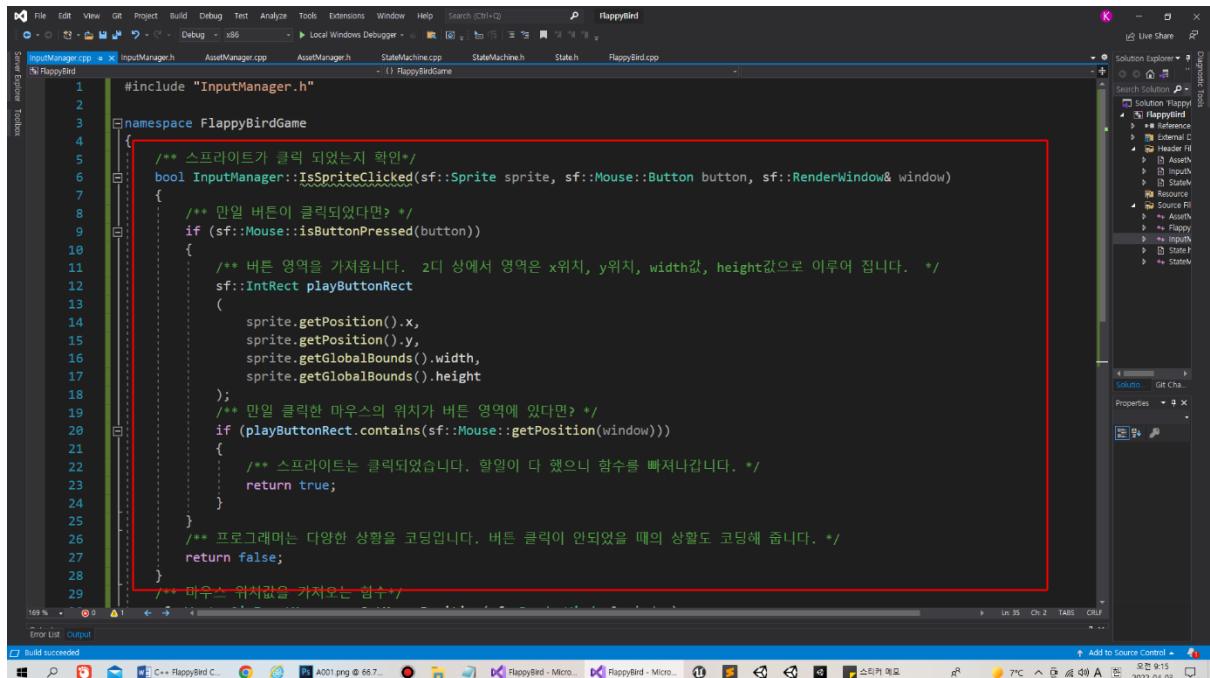
```
21 	/** 생성자 함수를 정의했습니다. */
22 	InputManager() {}
23 	/** 소멸자 함수를 정의했습니다. */
24 	~InputManager() {}
25 	/**
26 	* 생성자 함수의 원형을 선언했습니다.
27 	* 구현부에서 함수를 정의해 주어야 합니다.
28 	*/
29 	// InputManager();
30 	/** 스프라이트가 클릭 되었는지 확인하는 함수의 원형을 선언해 줍니다. */
31 	bool IsSpriteClicked(sf::Sprite object, sf::Mouse::Button button, sf::RenderWindow& window);
32 	/** 마우스 위치값을 가져오는 함수의 원형을 선언합니다. */
33 	sf::Vector2i Get.mousePosition(sf::RenderWindow& window);
34 	}
35 	}
36 };
```

구현부로 가서 필요한 라이브러리를 추가해 주도록 합니다.

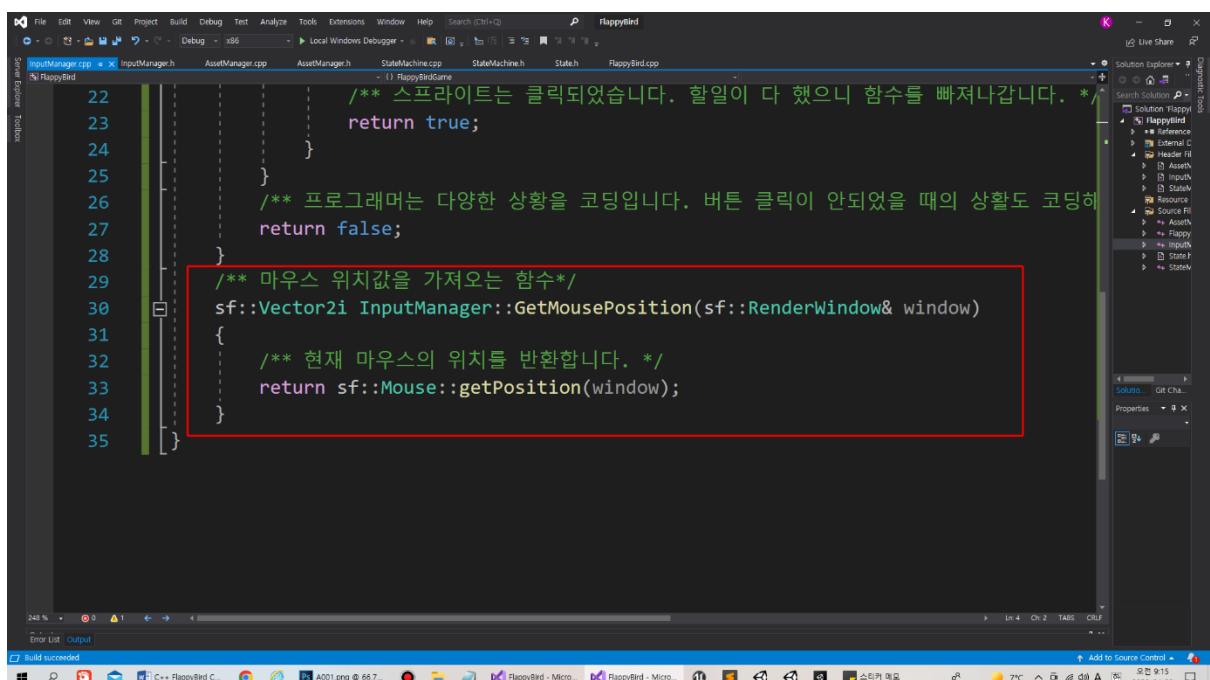


```
1 #include "InputManager.h"
2
3 namespace FlappyBirdGame
4 {
5 	/** 스프라이트가 클릭 되었는지 확인*/
6 	bool InputManager::IsSpriteClicked(sf::Sprite sprite, sf::RenderWindow window)
7 	{
8 		/** 만일 버튼이 클릭되었다면? */
9 		if (sf::Mouse::isButtonPressed(button))
10 		{
11 			/** 버튼 영역을 가져옵니다. 2디 상에서 영역은 x위치 */
12 			sf::IntRect playButtonRect
13 			(
14 				sprite.getPosition().x,
15 				sprite.getPosition().y,
```

구현부에서 구현해 주도록 합니다.

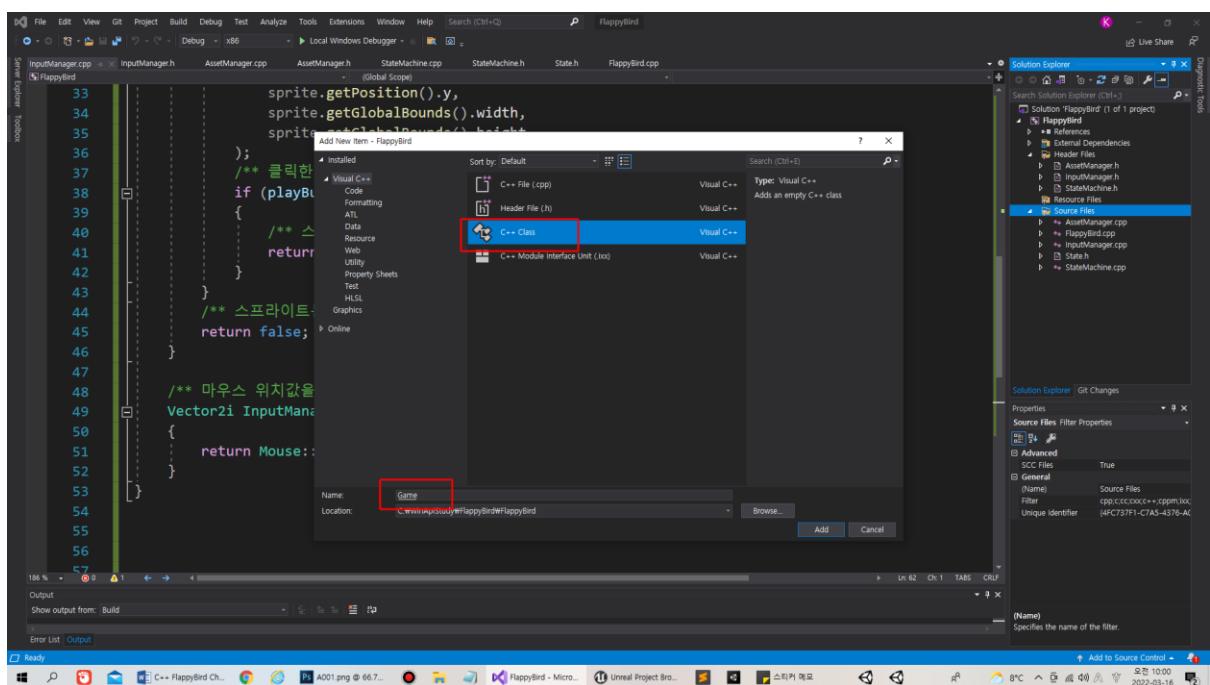
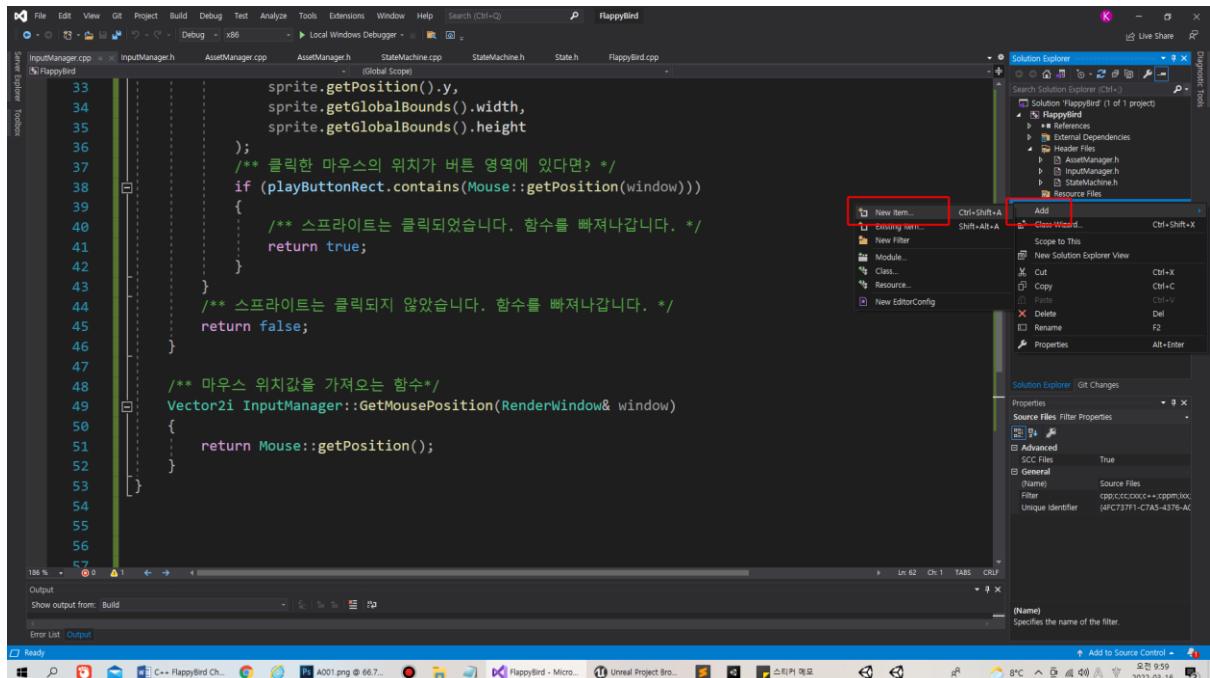


```
1 #include "InputManager.h"
2
3 namespace FlappyBirdGame
4 {
5     /** 스프라이트가 클릭 되었는지 확인*/
6     bool InputManager::IsSpriteClicked(sf::Sprite sprite, sf::Mouse::Button button, sf::RenderWindow& window)
7     {
8         /** 만일 버튼이 클릭되었다면? */
9         if (sf::Mouse::isButtonPressed(button))
10         {
11             /** 버튼 영역을 가져옵니다. 2d 상에서 영역은 x위치, y위치, width값, height값으로 이루어 집니다. */
12             sf::IntRect playButtonRect
13             (
14                 sprite.getPosition().x,
15                 sprite.getPosition().y,
16                 sprite.getGlobalBounds().width,
17                 sprite.getGlobalBounds().height
18             );
19             /** 만일 클릭한 마우스의 위치가 버튼 영역에 있다면? */
20             if (playButtonRect.contains(sf::Mouse::getPosition(window)))
21             {
22                 /** 스프라이트는 클릭되었습니다. 할일이 다 했으니 함수를 빠져나갑니다. */
23                 return true;
24             }
25         }
26         /** 프로그래머는 다양한 상황을 코딩입니다. 버튼 클릭이 안되었을 때의 상황도 코딩해 줍니다. */
27         return false;
28     }
29     /** 마우스 위치값을 가져오는 함수*/
30     sf::Vector2i InputManager::GetMousePosition(sf::RenderWindow& window)
31     {
32         /** 현재 마우스의 위치를 반환합니다. */
33         return sf::Mouse::getPosition(window);
34     }
35 }
```



```
22     /** 스프라이트는 클릭되었습니다. 할일이 다 했으니 함수를 빠져나갑니다. */
23     return true;
24 }
25 }
26 /** 프로그래머는 다양한 상황을 코딩입니다. 버튼 클릭이 안되었을 때의 상황도 코딩하
27     return false;
28 }
29 /** 마우스 위치값을 가져오는 함수*/
30 sf::Vector2i InputManager::GetMousePosition(sf::RenderWindow& window)
31 {
32     /** 현재 마우스의 위치를 반환합니다. */
33     return sf::Mouse::getPosition(window);
34 }
```

이제 Game 을 관리해줄 Game이라는 이름의 클래스를 만들어 주도록 합니다.



헤더 파일에서 필요한 라이브러리를 추가해 줍니다.

The screenshot shows the Microsoft Visual Studio IDE interface. The left pane displays the code for `Game.cpp` in the `FlappyBirdGame` namespace. The code includes #include statements for `string`, `sfml/Graphics`, `memory`, `StateMachine`, `AssetManager`, and `InputManager`. A red box highlights the `Game` header file inclusion. The right pane shows the `Solution Explorer` with the project structure, including `FlappyBird`, `Header Fil`, `Game`, `Input`, `State`, and `Resource` folders. The bottom status bar shows the build status as "Build succeeded".

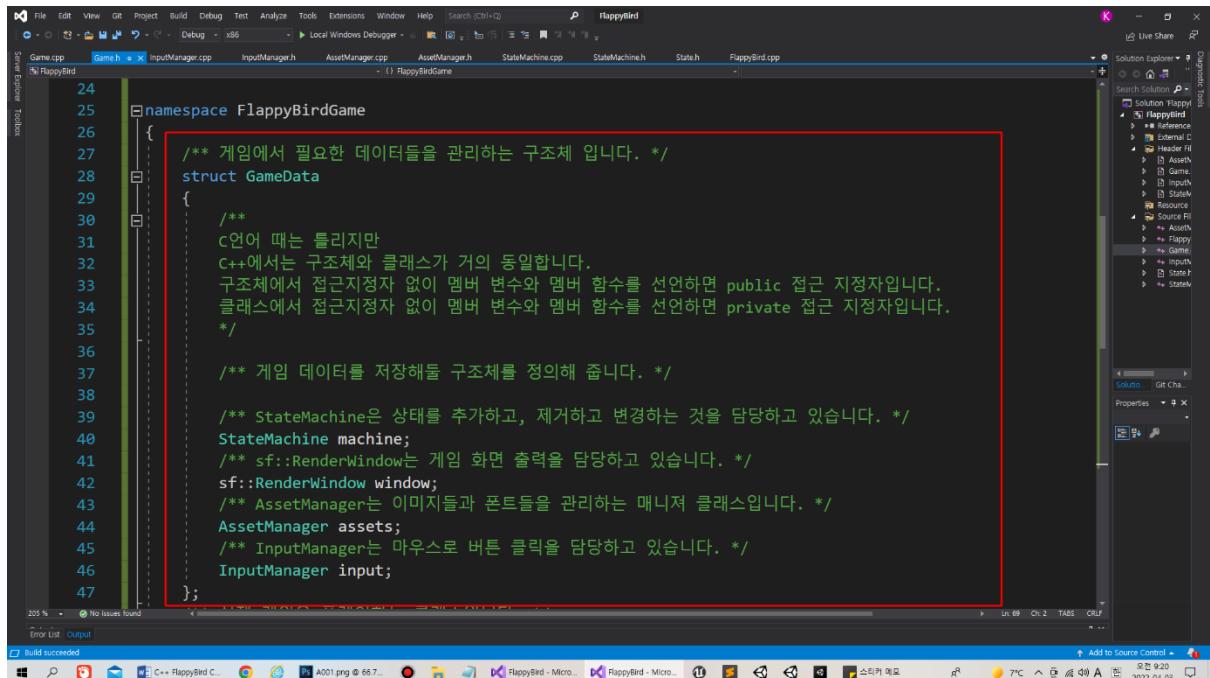
```
1 #pragma once
2
3 /* 문자열 관련 라이브러리 */
4 #include <string>
5 /* SFML Graphics 라이브러리 */
6 #include <SFML/Graphics.hpp>
7
8 /**
9  * 스마트 포인터 관련 라이브러리
10 * 여기저기서 접근이 가능한 실제 GameData를 shared_ptr로 메모리 관리 합니다.
11 */
12 #include <memory>
13 /* StateMachine에 접근하기 위한 헤더파일 */
14 #include "StateMachine.h"
15 /* AssetManager에 접근하기 위한 헤더파일 */
16 #include "AssetManager.h"
17 /* InputManager에 접근하기 위한 헤더파일 */
18 #include "InputManager.h"
19
20 /* C++ 표준 라이브러리를 사용합니다.*/
21 using namespace std;
22 /* SFML 라이브러리를 사용합니다.*/
23 using namespace sf;
24
25 /**
26  * 게임에서 필요한 데이터들을 관리하는 구조체입니다.
27 */
28
29 namespace FlappyBirdGame
30 {
31     /**
32      * ** 게임에서 필요한 데이터들을 관리하는 구조체입니다.
33     */
34 }
```

구현부에서 필요한 라이브러리를 추가해 줍니다.

The screenshot shows the Microsoft Visual Studio IDE interface. The left pane displays the `Game.h` header file, which includes the `Game` header. The right pane shows the `Game.cpp` implementation file, which includes the `Game` header and defines the `Game` class with its constructor and `Run` method. A red box highlights the `Game` header file inclusion. The bottom status bar shows the build status as "Build succeeded".

```
1 #include "Game.h"
2
3 namespace FlappyBirdGame
4 {
5     Game::Game(int width, int height, string title)
6     {
7         /**
8          * 생성자
9          */
10        _data->window.create(sf::VideoMode(width, height),
11                             title);
12
13        /**
14         * 생성자 통해서 객체생성하면 Run() 함수를 호출해서 게임 실행
15         */
16        this->Run();
17    }
18
19    /**
20     * 실제로 게임을 실행하는 멤버 함수입니다.
21     */
22    void Game::Run()
23    {
24    }
25 }
```

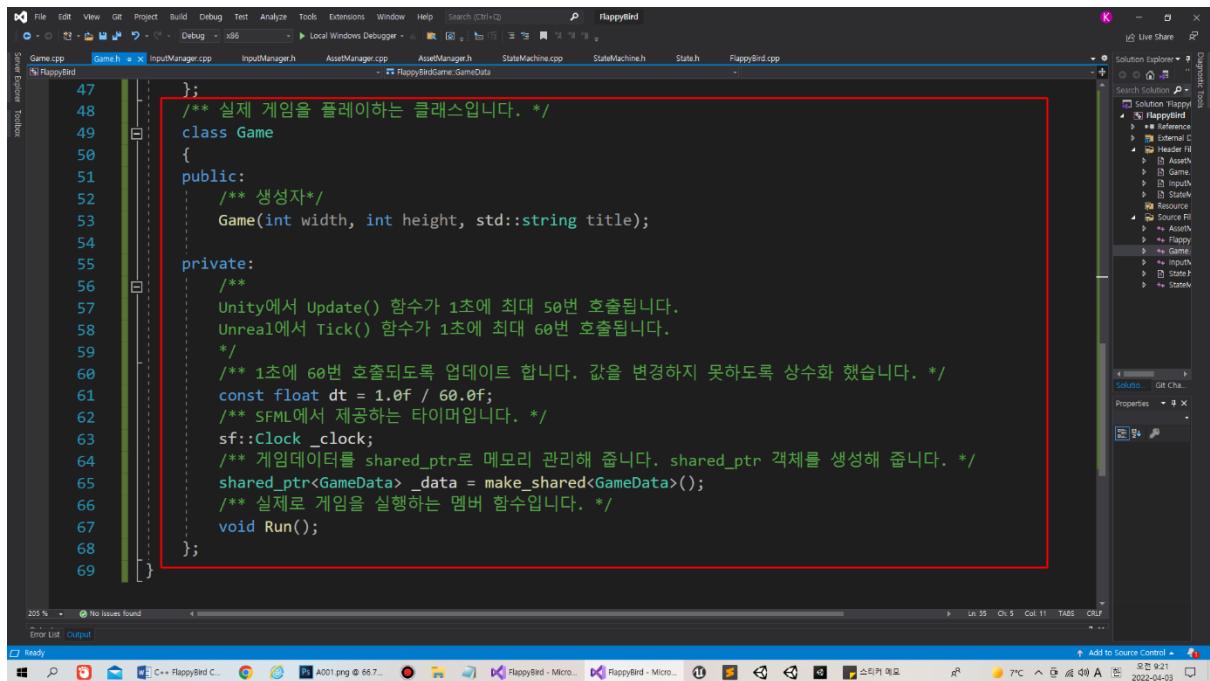
게임데이터를 저장해둘 구조체를 만들어 줍니다.



```
24
25     namespace FlappyBirdGame
26     {
27         /** 게임에서 필요한 데이터들을 관리하는 구조체입니다. */
28         struct GameData
29         {
30             /**
31             C언어 때는 틀리지만
32             C++에서는 구조체와 클래스가 거의 동일합니다.
33             구조체에서 접근지정자 없이 멤버 변수와 멤버 함수를 선언하면 public 접근 지정자입니다.
34             클래스에서 접근지정자 없이 멤버 변수와 멤버 함수를 선언하면 private 접근 지정자입니다.
35             */
36
37             /** 게임 데이터를 저장해둘 구조체를 정의해 줍니다. */
38
39             /** StateMachine은 상태를 추가하고, 제거하고 변경하는 것을 담당하고 있습니다. */
40             StateMachine machine;
41             /** sf::RenderWindow는 게임 화면 출력을 담당하고 있습니다. */
42             sf::RenderWindow window;
43             /** AssetManager는 이미지들과 폰트들을 관리하는 매니저 클래스입니다. */
44             AssetManager assets;
45             /** InputManager는 마우스로 버튼 클릭을 담당하고 있습니다. */
46             InputManager input;
47         };

```

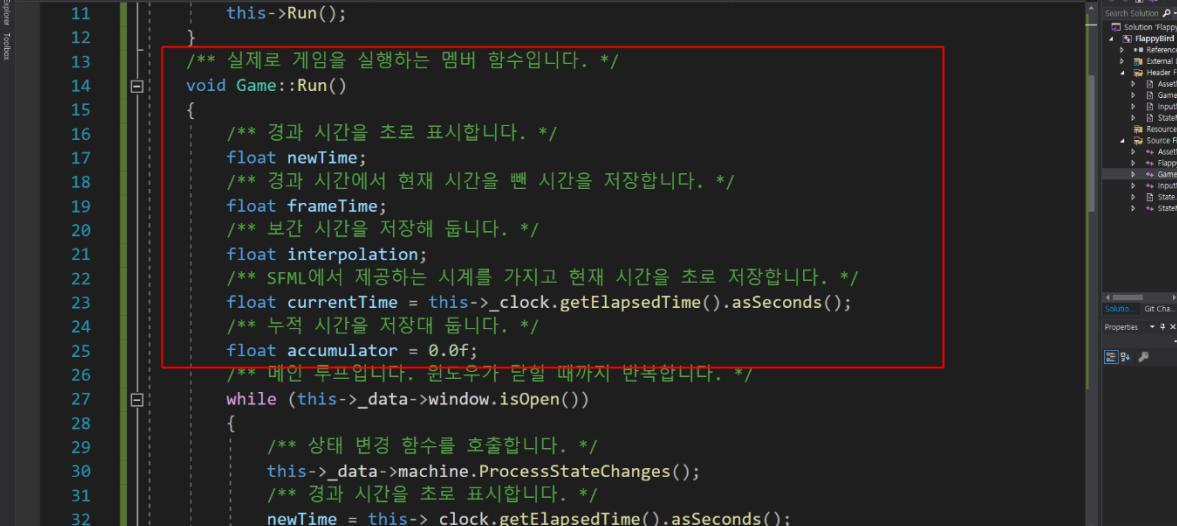
클래스를 정의해 주도록 합니다.



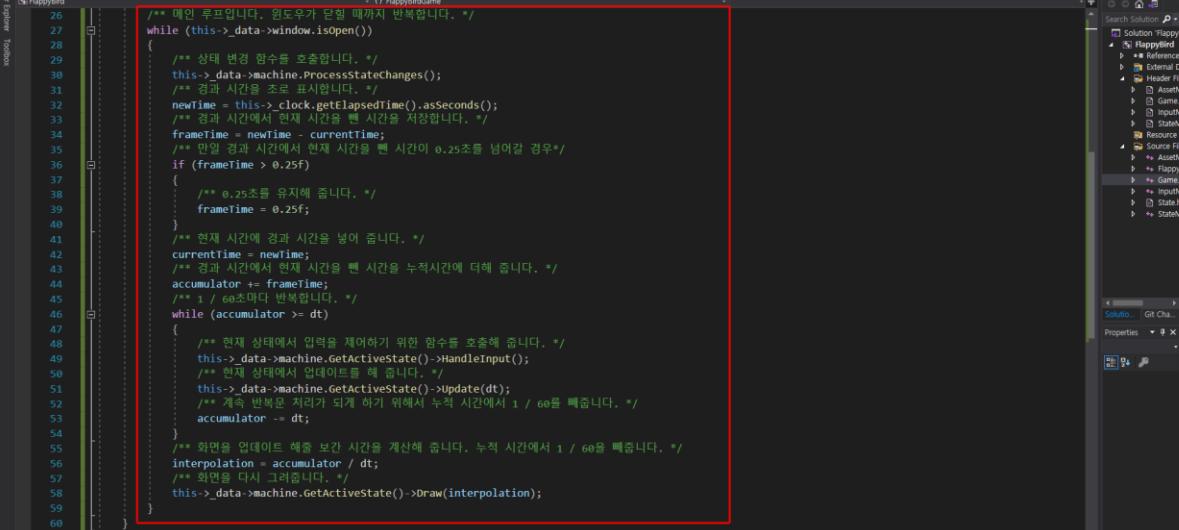
```
47
48     /**
49     * 실제 게임을 플레이하는 클래스입니다.
50     */
51     class Game
52     {
53     public:
54         /**
55         * 생성자
56         */
57         Game(int width, int height, std::string title);
58
59     private:
60         /**
61         * Unity에서 Update() 함수가 1초에 최대 50번 호출됩니다.
62         * Unreal에서 Tick() 함수가 1초에 최대 60번 호출됩니다.
63         */
64         const float dt = 1.0f / 60.0f;
65         /**
66         * SFML에서 제공하는 타이머입니다.
67         */
68         sf::Clock _clock;
69         /**
70         * 게임데이터를 shared_ptr로 메모리 관리해 줍니다. shared_ptr 객체를 생성해 줍니다.
71         */
72         shared_ptr<GameData> _data = make_shared<GameData>();
73         /**
74         * 실제로 게임을 실행하는 멤버 함수입니다.
75         */
76         void Run();
77     };

```

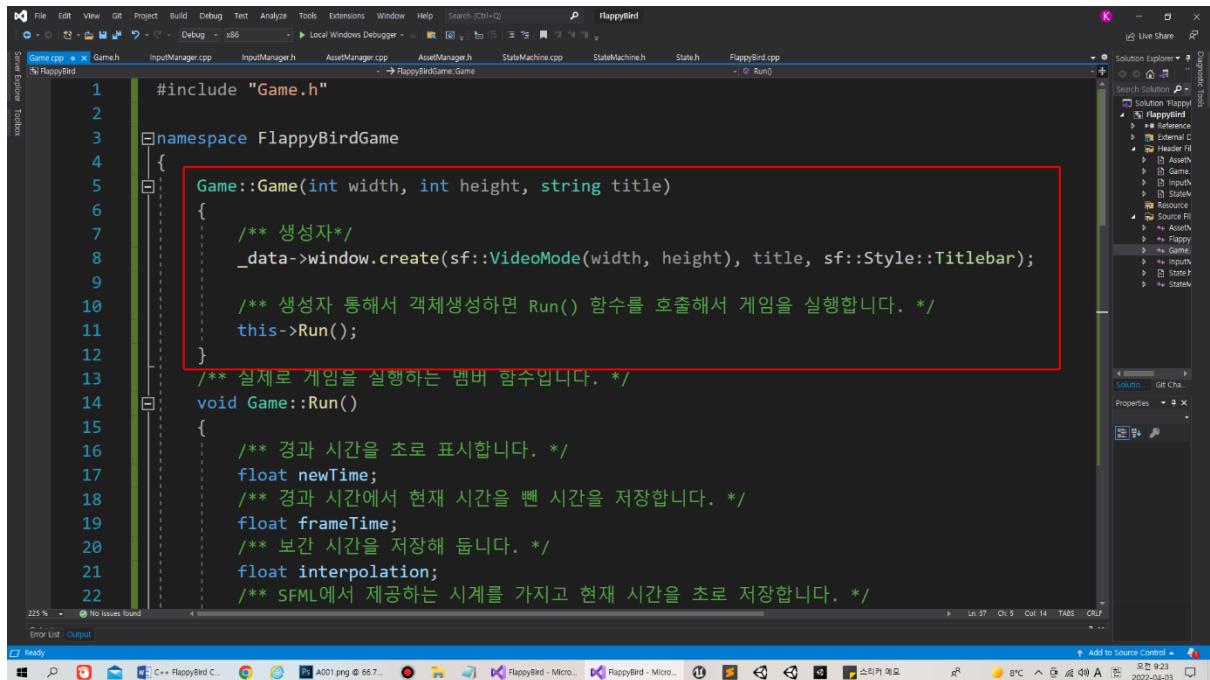
구현부에서 구현해 주도록 합니다.



```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+F2) Live Share
Game.cpp Game.h InputManager.cpp InputManager.h AssetManager.cpp AssetManager.h StateMachine.cpp StateMachine.h State.h FlappyBird.cpp
Server Explorer Tools
11     this->Run();
12 }
13 /* 실제로 게임을 실행하는 멤버 함수입니다. */
14 void Game::Run()
15 {
16     /* 경과 시간을 초로 표시합니다. */
17     float newTime;
18     /* 경과 시간에서 현재 시간을 뺀 시간을 저장합니다. */
19     float frameTime;
20     /* 보간 시간을 저장해 둡니다. */
21     float interpolation;
22     /* SFML에서 제공하는 시계를 가지고 현재 시간을 초로 저장합니다. */
23     float currentTime = this->_clock.getElapsedTime().asSeconds();
24     /* 누적 시간을 저장해 둡니다. */
25     float accumulator = 0.0f;
26     /* 메인 루프입니다. 창이 닫힐 때까지 반복합니다. */
27     while (this->_data->window.isOpen())
28     {
29         /* 상태 변경 함수를 호출합니다. */
30         this->_data->machine.ProcessStateChanges();
31         /* 경과 시간을 초로 표시합니다. */
32         newTime = this->_clock.getElapsedTime().asSeconds();
33     }
34 }
```

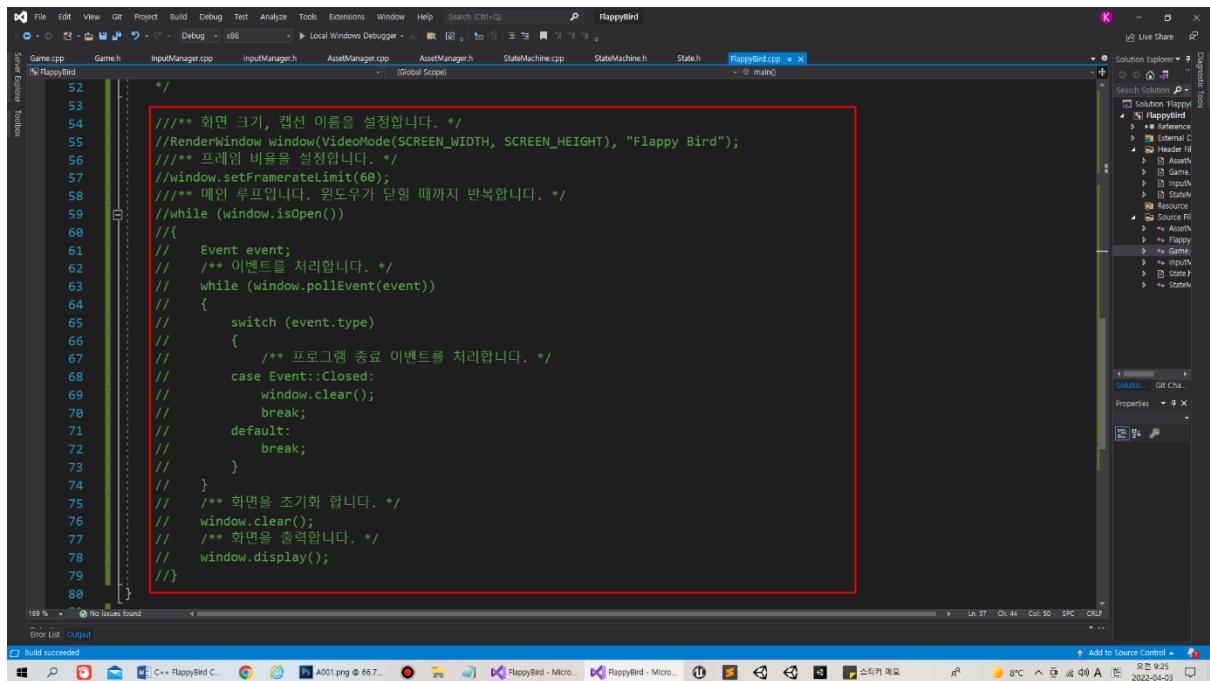


```
/* 메인 후프입니다. 윈도우가 닫힐 때까지 반복합니다. */
while (this -> data -> window.isOpen())
{
    /* 상태 변경 함수를 호출합니다. */
    this -> data -> machine.ProcessStateChanges();
    /* 경과 시간을 초기 표시합니다. */
    newTime = this -> clock.getLapsedTime().asSeconds();
    /* 경과 시간에서 현재 시간을 뺀 시간을 저장합니다. */
    frameTime = newTime - currentTime;
    /* 만약 경과 시간에서 현재 시간을 뺀 시간이 0.25초를 넘어갈 경우 */
    if (frameTime > 0.25f)
    {
        /* 0.25초를 유지해 줍니다. */
        frameTime = 0.25f;
    }
    /* 현재 시간에 경과 시간을 넣어 줍니다. */
    currentTime = newTime;
    /* 경과 시간에서 현재 시간을 뺀 시간을 누적시간에 더해 줍니다. */
    accumulator += frameTime;
    /* 1 / 60초마다 반복합니다. */
    while (accumulator > dt)
    {
        /* 현재 상태에서 입력을 제어하기 위한 함수를 호출해 줍니다. */
        this -> data -> machine.GetActiveState() -> HandleInput();
        /* 현재 상태에서 업데이트를 해 줍니다. */
        this -> data -> machine.GetActiveState() -> Update(dt);
        /* 계속 반복문 처리가 되게 하기 위해서 누적 시간에서 1 / 60을 빼줍니다. */
        accumulator -= dt;
    }
    /* 화면을 업데이트 해줄 보간 시간을 계산해 줍니다. 누적 시간에서 1 / 60을 빼줍니다. */
    interpolation = accumulator / dt;
    /* 화면을 다시 그리줍니다. */
    this -> data -> machine.GetActiveState() -> Draw(interpolation);
}
}
```



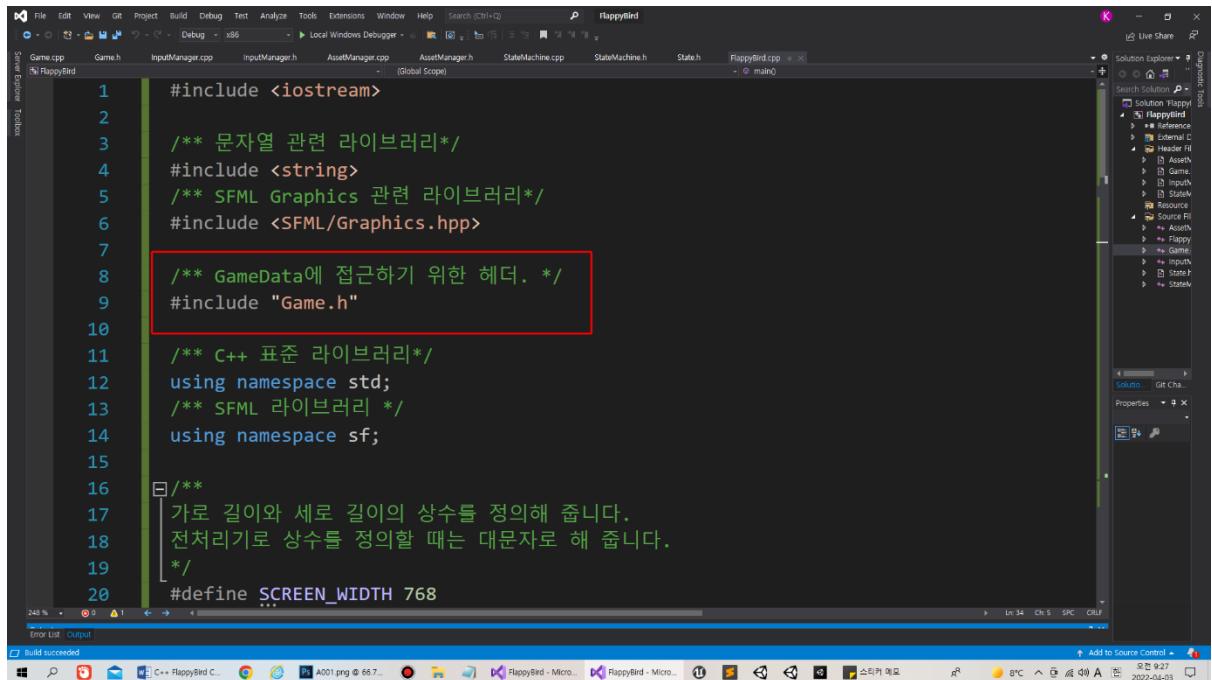
```
1 #include "Game.h"
2
3 namespace FlappyBirdGame
4 {
5     Game::Game(int width, int height, string title)
6     {
7         /* 생성자 */
8         _data->window.create(sf::VideoMode(width, height), title, sf::Style::Titlebar);
9
10        /* 생성자 통해서 객체생성하면 Run() 함수를 호출해서 게임을 실행합니다. */
11        this->Run();
12    }
13
14    /* 실제로 게임을 실행하는 멤버 함수입니다. */
15    void Game::Run()
16    {
17        /* 경과 시간을 초로 표시합니다. */
18        float newTime;
19        /* 경과 시간에서 현재 시간을 뺀 시간을 저장합니다. */
20        float frameTime;
21        /* 보간 시간을 저장해 둡니다. */
22        float interpolation;
23        /* SFML에서 제공하는 시계를 가지고 현재 시간을 초로 저장합니다. */
24    }
25 }
```

메인 함수의 내용을 주석처리 해 줍니다.

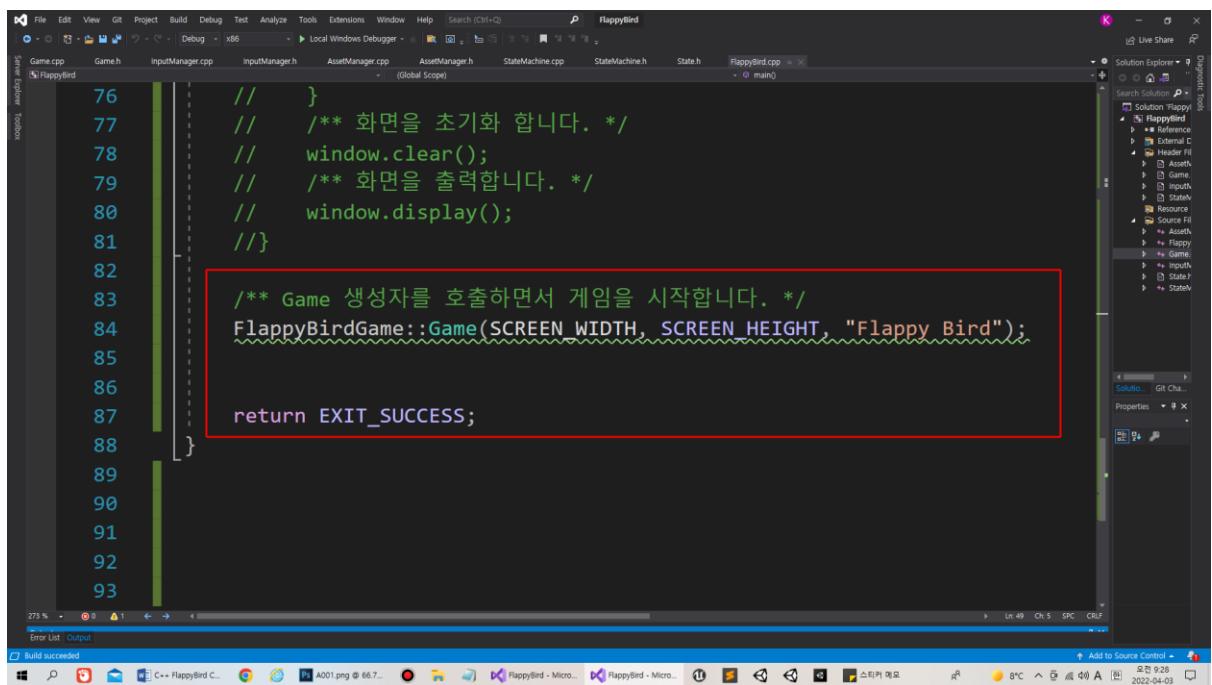


```
52 */
53
54     /* 화면 크기, 캔선 이름을 설정합니다. */
55     //RenderWindow window(VideoMode(SCREEN_WIDTH, SCREEN_HEIGHT), "Flappy Bird");
56     /* 프레임 비율을 설정합니다. */
57     //window.setFramerateLimit(60);
58     /* 메인 루프입니다. 원도우가 닫힐 때까지 반복합니다. */
59     //while (window.isOpen())
60     //{
61         // Event event;
62         /* 이벤트를 처리합니다. */
63         // while (window.pollEvent(event))
64         //{
65             // switch (event.type)
66             //{
67                 /* 프로그램 종료 이벤트를 처리합니다. */
68                 case Event::Closed:
69                     window.clear();
70                     break;
71                 default:
72                     break;
73             }
74         }
75         /* 화면을 초기화 합니다. */
76         window.clear();
77         /* 화면을 출력합니다. */
78         window.display();
79     //}
80 }
```

내용을 추가해 줍니다.



```
1 #include <iostream>
2
3 /** 문자열 관련 라이브러리*/
4 #include <string>
5 /** SFML Graphics 관련 라이브러리*/
6 #include <SFML/Graphics.hpp>
7
8 /** GameData에 접근하기 위한 헤더. */
9 #include "Game.h"
10
11 /** C++ 표준 라이브러리*/
12 using namespace std;
13 /** SFML 라이브러리 */
14 using namespace sf;
15
16 /**
17  * 가로 길이와 세로 길이의 상수를 정의해 줍니다.
18  * 전처리기로 상수를 정의할 때는 대문자로 해 줍니다.
19  */
20 #define SCREEN_WIDTH 768
```



```
76 }
77 /**
78  * 화면을 초기화 합니다.
79  * window.clear();
80  * 화면을 출력합니다.
81  * window.display();
82 */
83
84 /**
85  * Game 생성자를 호출하면서 게임을 시작합니다.
86  */
87 FlappyBirdGame::Game(SCREEN_WIDTH, SCREEN_HEIGHT, "Flappy_Bird");
88
89
90
91
92
93 }
```