|  |
| --- |
| [Unity3D] 고스톱 게임 만들기 - 3 (카드 뒤집기 로직 구현) |

**2.4 카드 뒤집기**

카드를 낸 후 다음 순서는 가운데 카드를 뒤집는 것입니다.

뒤집은 카드에 대해서도 바닥 카드와 숫자를 비교하여 같은 숫자의 카드가 있으면 플레이어가 가져갈 수 있습니다.

뒤집기 처리부분의 소스코드를 보겠습니다.

·미리보기 | 소스복사·

1. PLAYER\_SELECT\_CARD\_RESULT flip\_deck\_card(byte card\_number\_from\_player)
2. {
3. this.card\_from\_deck = pop\_front\_card();
5. List<CCard> same\_cards = this.floor\_manager.get\_cards(this.card\_from\_deck.number);
6. if (same\_cards != null)
7. {
8. this.same\_card\_count\_with\_deck = (byte)same\_cards.Count;
9. }
10. else
11. {
12. this.same\_card\_count\_with\_deck = 0;
13. }
15. //UnityEngine.Debug.Log(string.Format("flipped card {0},  same card count {1}",
16. //  this.card\_from\_deck.number, this.same\_card\_count\_with\_deck));
18. switch (this.same\_card\_count\_with\_deck)
19. {
20. case 0:
21. if (card\_number\_from\_player == this.card\_from\_deck.number)
22. {
23. // 쪽.
24. this.flipped\_card\_event\_type.Add(CARD\_EVENT\_TYPE.KISS);
26. this.cards\_to\_give\_player.Clear();
27. this.cards\_to\_give\_player.Add(this.card\_from\_player);
28. this.cards\_to\_give\_player.Add(this.card\_from\_deck);
30. take\_cards\_from\_others(1);
31. }
32. break;
34. case 1:
35. {
36. if (card\_number\_from\_player == this.card\_from\_deck.number)
37. {
38. // 뻑.
39. this.flipped\_card\_event\_type.Add(CARD\_EVENT\_TYPE.PPUK);
40. get\_current\_player().plus\_ppuk\_count();
42. // 플레이어에게 주려던 카드를 모두 취소한다.
43. this.cards\_to\_give\_player.Clear();
44. }
45. else
46. {
47. this.cards\_to\_give\_player.Add(this.card\_from\_deck);
48. this.cards\_to\_give\_player.Add(same\_cards[0]);
49. }
50. }
51. break;
53. case 2:
54. {
55. if (this.card\_from\_deck.number == card\_number\_from\_player)
56. {
57. // 따닥.
58. this.flipped\_card\_event\_type.Add(CARD\_EVENT\_TYPE.DDADAK);
60. // 플레이어가 4장 모두 가져간다.
61. this.cards\_to\_give\_player.Clear();
62. for (int i = 0; i < same\_cards.Count; ++i)
63. {
64. this.cards\_to\_give\_player.Add(same\_cards[i]);
65. }
66. this.cards\_to\_give\_player.Add(this.card\_from\_deck);
67. this.cards\_to\_give\_player.Add(this.card\_from\_player);
69. take\_cards\_from\_others(2);
70. }
71. else
72. {
73. if (same\_cards[0].pae\_type != same\_cards[1].pae\_type)
74. {
75. // 뒤집었는데 타입이 다른 카드 두장과 같다면 한장을 선택하도록 한다.
76. this.target\_cards\_to\_choice.Clear();
77. for (int i = 0; i < same\_cards.Count; ++i)
78. {
79. this.target\_cards\_to\_choice.Add(same\_cards[i]);
80. }
82. this.expected\_result\_type = PLAYER\_SELECT\_CARD\_RESULT.CHOICE\_ONE\_CARD\_FROM\_DECK;
83. return PLAYER\_SELECT\_CARD\_RESULT.CHOICE\_ONE\_CARD\_FROM\_DECK;
84. }
86. // 카드 타입이 같다면 첫번째 카드를 선택해 준다.
87. this.cards\_to\_give\_player.Add(this.card\_from\_deck);
88. this.cards\_to\_give\_player.Add(same\_cards[0]);
89. }
90. }
91. break;
93. case 3:
94. // 플레이어가 4장 모두 가져간다.
95. for (int i = 0; i < same\_cards.Count; ++i)
96. {
97. this.cards\_to\_give\_player.Add(same\_cards[i]);
98. }
99. this.cards\_to\_give\_player.Add(this.card\_from\_deck);
101. //todo:자뻑이라면 두장.
102. take\_cards\_from\_others(1);
103. break;
104. }
106. return PLAYER\_SELECT\_CARD\_RESULT.COMPLETED;
107. }

뒤집은 카드의 처리는 플레이어가 낸 카드의 처리와 비슷합니다.

뒤집을때만 발생할 수 있는 규칙으로는 쪽, 뻑, 따닥이 있습니다.

쪽은 낸 카드와 뒤집은 카드의 번호가 똑같을 때

뻑은 뒤집은 결과 같은 번호의 카드가 세장 겹쳤을 때

따닥은 뒤집은 결과 같은 번호의 카드가 네장 겹쳤을 때를 말합니다.

**2.5 먹을 카드와 뺏을 카드 정하기**

카드 뒤집기 까지 처리가 완료되면 플레이어가 먹을 카드와 바닥에 내려놓을 카드를 선별하는 작업을 진행합니다.

이 부분의 소스코드입니다.

·미리보기 | 소스복사·

1. /// <summary>
2. /// 카드를 뒤집은 후 처리해야할 내용들을 진행한다.
3. /// </summary>
4. /// <param name="player\_index"></param>
5. void after\_flipped\_card(byte player\_index)
6. {
7. // 플레이어가 가져갈 카드와 바닥에 내려놓을 카드를 처리한다.
8. give\_floor\_cards\_to\_player(player\_index);
9. sort\_player\_pae();
10. calculate\_players\_score();

13. // 폭탄으로 뒤집는 경우에는 플레이어가 낸 카드가 없으므로 처리를 건너 뛴다.
14. if (this.card\_from\_player != null)
15. {
16. // 플레이어가 가져갈 카드중에 냈던 카드가 포함되어 있지 않으면 바닥에 내려 놓는다.
17. bool is\_exist\_player = this.cards\_to\_give\_player.Exists(obj => obj.is\_same(
18. this.card\_from\_player.number,
19. this.card\_from\_player.pae\_type,
20. this.card\_from\_player.position));
21. if (!is\_exist\_player)
22. {
23. this.floor\_manager.puton\_card(this.card\_from\_player);
24. this.cards\_to\_floor.Add(this.card\_from\_player);
25. }
26. }

29. // 뒤집은 카드에 대해서도 같은 방식으로 처리한다.
30. bool is\_exist\_deck\_card = this.cards\_to\_give\_player.Exists(obj => obj.is\_same(
31. this.card\_from\_deck.number,
32. this.card\_from\_deck.pae\_type,
33. this.card\_from\_deck.position));
34. if (!is\_exist\_deck\_card)
35. {
36. this.floor\_manager.puton\_card(this.card\_from\_deck);
37. this.cards\_to\_floor.Add(this.card\_from\_deck);
38. calculate\_players\_score();
39. }
41. // 싹쓸이 체크.
42. if (this.floor\_manager.is\_empty())
43. {
44. this.flipped\_card\_event\_type.Add(CARD\_EVENT\_TYPE.CLEAN);
45. take\_cards\_from\_others(1);
46. }
47. }

플레이어가 낸 카드와 뒤집은 카드의 로직을 통해

어떤 카드를 가져가야 할지 임시로 카드들을 담아놓은 변수가 있습니다.

cards\_to\_give\_player라는 리스트에 담아 놓았는데 이 리스트와

플레이어가 낸 카드를 비교하여 가져갈지 바닥에 내려놓을지를 결정 합니다.

모든 정리가 끝난 후 바닥에 한 장의 카드도 남지 않게 된다면

‘싹쓸이’라는 룰을 적용시켜 상대방으로부터 카드를 한 장 뺏어올 수 있습니다.

**2.6 점수 계산**

가져올 카드와 내려놓을 카드를 결정한 후에는 플레이어들의 점수를 계산 합니다.

점수 계산 코드는 CPlayerAgent라는 클래스에 구현되어 있습니다.

먼저 소스코드를 보겠습니다.

·미리보기 | 소스복사·

1. public void calculate\_score()
2. {
3. this.score = 0;
4. this.score += get\_score\_by\_type(PAE\_TYPE.PEE);
5. this.score += get\_score\_by\_type(PAE\_TYPE.TEE);
6. this.score += get\_score\_by\_type(PAE\_TYPE.YEOL);
7. this.score += get\_score\_by\_type(PAE\_TYPE.KWANG);
9. // 고도리
10. byte godori\_count = get\_card\_count(PAE\_TYPE.YEOL, CARD\_STATUS.GODORI);
11. if (godori\_count == 3)
12. {
13. this.score += 5;
14. //UnityEngine.Debug.Log("Godori 5 score");
15. }
17. // 홍단, 초단, 청단
18. byte cheongdan\_count = get\_card\_count(PAE\_TYPE.TEE, CARD\_STATUS.CHEONG\_DAN);
19. byte hongdan\_count = get\_card\_count(PAE\_TYPE.TEE, CARD\_STATUS.HONG\_DAN);
20. byte chodan\_count = get\_card\_count(PAE\_TYPE.TEE, CARD\_STATUS.CHO\_DAN);
21. if (cheongdan\_count == 3)
22. {
23. this.score += 3;
24. //UnityEngine.Debug.Log("Cheongdan 3 score");
25. }
27. if (hongdan\_count == 3)
28. {
29. this.score += 3;
30. //UnityEngine.Debug.Log("Hongdan 3 score");
31. }
33. if (chodan\_count == 3)
34. {
35. this.score += 3;
36. //UnityEngine.Debug.Log("Chodan 3 score");
37. }
39. //UnityEngine.Debug.Log(string.Format("[SCORE] player {0},  score {1}", this.player\_index, this.score));
40. }

피, 띠, 열끗, 광의 점수를 각각 계산한 뒤

고도리, 청단, 홍단, 초단을 달성했을 경우의 점수를 더합니다.

고도리, 청단, 홍단, 초단은 특정 카드를 모았을 때 일정 점수를 획득할 수 있는 규칙입니다.

여기 까지 고스톱의 주요 로직 구현을 알아 봤습니다.

다음 강좌에서는 게임방과 유저의 입력을 연결시켜 주는 부분과

클라이언트 그래픽 처리 부분을 살펴보겠습니다.

감사합니다.