**[Unity3D] 고스톱 게임 만들기 - 6 (화면 구성)**

\*\*\* 광고 \*\*\*

소스코드 저장소를 네이버 개발자 센터에서 github로 변경하였습니다.

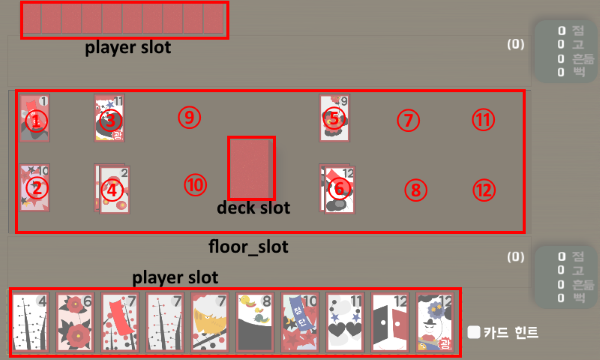
기존의 svn으로도 체크아웃 받으실 수 있지만 더이상 새로 올라오는 커밋은 없을것이기 때문에

앞으로는 git을 이용하여 github에서 받아주시면 감사하겠습니다.

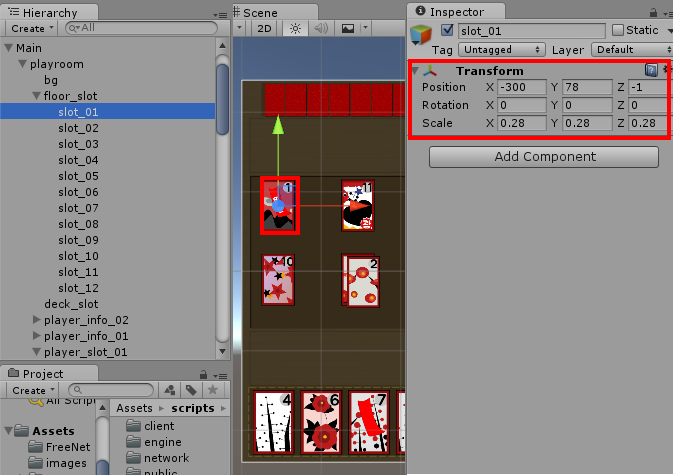
**소스코드 링크 :**[**https://github.com/sunduk/freegostop**](https://github.com/sunduk/freegostop)

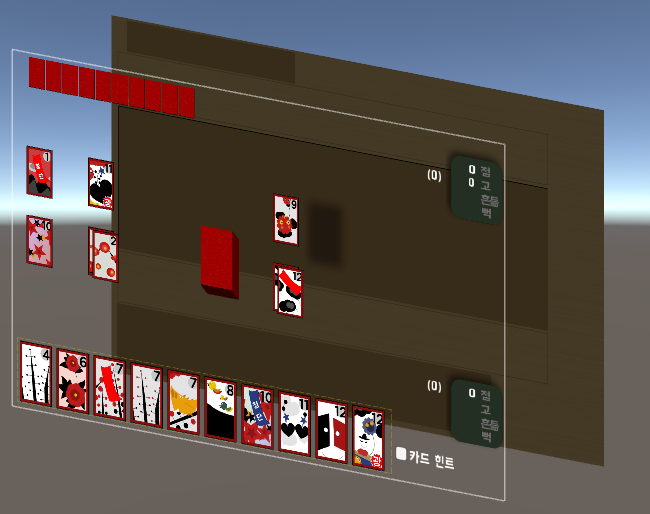
이번 강좌에서는 클라이언트 게임 화면 구성과 그 클래스에 대해 살펴보겠습니다.

변수 하나하나에 대한 설명 보다는 전체적으로 어떻게 구성되어 있는지 대강 파악하는 수준으로 진행하도록 하겠습니다.

  
카드 포지션 슬롯 구성

게임 화면은 위 그림처럼 배경 이미지와 카드가 위치할 슬롯들로 구성 됩니다.

  
슬롯 오브젝트는 Transform정보만 갖고 있다.

  
슬롯 뒤에는 게임의 보드판 이미지가 깔려 있다.

슬롯과 보드판을 처리하는 클래스는 class CPlayRoomUI입니다.

씬의 Main/playroom 오브젝트에 연결되어 있어 게임 진입시 자동으로 활성화 됩니다.

이곳에서 카드의 움직임 처리가 구현되기 때문에 클라이언트의 핵심적인 클래스라고 말할 수 있습니다.

멤버 변수

·미리보기 | 소스복사·

1. // 원본 이미지들.
3. Sprite back\_image;




9. // 각 슬롯의 좌표 객체.
11. [SerializeField]
13. Transform floor\_slot\_root;
15. List<Vector3> floor\_slot\_position;


19. [SerializeField]
21. Transform deck\_slot;


25. [SerializeField]
27. CPlayerCardPosition[] player\_card\_positions;




33. // 카드 객체.
35. List<CCardPicture> total\_card\_pictures;


39. CCardCollision card\_collision\_manager;


43. // 자리별 카드 스케일.
45. readonly Vector3 SCALE\_TO\_FLOOR = new Vector3(0.8f, 0.8f, 0.8f);
47. readonly Vector3 SCALE\_TO\_OTHER\_HAND = new Vector3(0.5f, 0.5f, 0.5f);
49. readonly Vector3 SCALE\_TO\_MY\_HAND = new Vector3(1.0f, 1.0f, 1.0f);


53. readonly Vector3 SCALE\_TO\_OTHER\_FLOOR = new Vector3(0.6f, 0.6f, 0.6f);
55. readonly Vector3 SCALE\_TO\_MY\_FLOOR = new Vector3(0.6f, 0.6f, 0.6f);




61. // 게임 플레이에 사용되는 객체들.
63. byte player\_me\_index;
65. List<CVisualFloorSlot> floor\_ui\_slots;
67. // 가운데 쌓여있는 카드 객체.
69. Stack<CCardPicture> deck\_cards;
71. List<CPlayerHandCardManager> player\_hand\_card\_manager;
73. // 플레이어가 먹은 카드 객체.
75. List<CPlayerCardManager> player\_card\_manager;
77. List<CPlayerInfoSlot> player\_info\_slots;


81. CCardManager card\_manager;


85. Queue<CPacket> waiting\_packets;




91. // 효과 관련 객체들.
93. GameObject ef\_focus;




99. // 테스트용 변수들.
101. bool is\_test\_mode = false;
103. byte test\_auto\_slot\_index;

Sprite back\_image 는 카드 뒷면을 나타내는 스프라이트입니다. 덱에 쌓인 카드나 상대방 카드를 출력할 때 이 이미지를 사용하여 출력하게 됩니다.

단순히 뒷면만 보여지는 것 뿐만 아니라 실제로 클라이언트 로직 내에서 상대방이

무슨 카드를 가지고 있는지조차 알 수 없도록 구현되어 있습니다.

·미리보기 | 소스복사·

1. // 각 슬롯의 좌표 객체.
3. [SerializeField]
5. Transform floor\_slot\_root;
7. List<Vector3> floor\_slot\_position;


11. [SerializeField]
13. Transform deck\_slot;


17. [SerializeField]
19. CPlayerCardPosition[] player\_card\_positions;

앞에서 봤던 슬롯 구성 화면에서 각 슬롯 오브젝트와 매칭되는 변수들입니다.

각 슬롯의 위치 정보들을 변수에 저장해 놓은 뒤 카드 움직임 구현시 사용하게 됩니다.

·미리보기 | 소스복사·

1. // 카드 객체.
3. List<CCardPicture> total\_card\_pictures;
5. CCardCollision card\_collision\_manager;

total\_card\_pictures는 CCardPicture라는 카드 객체를 갖고 있는 리스트입니다.

CCardPicture객체는 카드의 수치 데이터와 이미지를 함께 갖고 있어 어떤 카드가 어떤 이미지를 나타내는지 한번에 알 수 있게 구성되어 있습니다.

·미리보기 | 소스복사·

1. // 자리별 카드 스케일.
2. readonly Vector3 SCALE\_TO\_FLOOR = new Vector3(0.8f, 0.8f, 0.8f);
3. readonly Vector3 SCALE\_TO\_OTHER\_HAND = new Vector3(0.5f, 0.5f, 0.5f);
4. readonly Vector3 SCALE\_TO\_MY\_HAND = new Vector3(1.0f, 1.0f, 1.0f);
5. readonly Vector3 SCALE\_TO\_OTHER\_FLOOR = new Vector3(0.6f, 0.6f, 0.6f);
6. readonly Vector3 SCALE\_TO\_MY\_FLOOR = new Vector3(0.6f, 0.6f, 0.6f);

슬롯별 카드 스케일 값을 나타냅니다.

내 카드와 상대방 카드 또는 바닥에 놓인 카드들의 스케일이 전부 다르게 적용되도록 설계하였기 때문에 각 위치에 맞는 스케일 값을 보관해 놓을 필요가 있습니다.

이 값을 이용하여 카드가 움직일 때 어느 위치로 갈 것인지에 따라 스케일 값을 다르게 넣어줄 수 있습니다.

·미리보기 | 소스복사·

1. // 게임 플레이에 사용되는 객체들.
3. byte player\_me\_index;
5. List<CVisualFloorSlot> floor\_ui\_slots;
7. // 가운데 쌓여있는 카드 객체.
9. Stack<CCardPicture> deck\_cards;
11. List<CPlayerHandCardManager> player\_hand\_card\_manager;
13. // 플레이어가 먹은 카드 객체.
15. List<CPlayerCardManager> player\_card\_manager;
17. List<CPlayerInfoSlot> player\_info\_slots;


21. CCardManager card\_manager;


25. Queue<CPacket> waiting\_packets;




31. // 효과 관련 객체들.
33. GameObject ef\_focus;

게임 진행중 필요한 변수들입니다. 변수명과 주석을 통해 그 의미를 쉽게 파악하실 수 있을 겁니다.

여기까지 CPlayRoomUI클래스의 멤버 변수까지 파악해 봤습니다.  
다음 장에서는 카드의 움직임이 어떻게 구현되는지 알아보겠습니다.