Unity 3D RPG

02. Entity Class 정의와 UI 설정

2016-10-10

Created By. ChoA.

Index

- Constants Script
- Entity Class 정의
- Player Class 수정
- Base UI 설정

Constants Script

- Constants Script 생성 및 작성
- Player Script 수정

Constants Script

Constants Script 생성 및 작성

- 전역으로 사용될 변수들을 관리하는 스크립트 생성 및 작성
  - Project View - 마우스 오른쪽 클릭 - Create - C# Script
  - 생성한 스크립트의 이름을 "Constants"로 변경

```
1  public enum ENTITY_STATE
2  {
3      IDLE=0, MOVE, ATTACK, HIT, CASTING, DIE,
4  }
5  public enum ENTITY_CLASS
6  {
7      NOOB=0, MAGE, WARRIOR, ARCHER,
8  }
```

Constants Script

- Player Script 수정

```
1  using UnityEngine;
2  using System.Collections;
3
4  public enum PLAYER_STATE { IDLE=0, MOVE }
5
6  public class Player : MonoBehaviour
7  {
8      private Animator        anim;
9      private PLAYER_STATE    player_state;
10     private float           idle_time;
11
12     private Vector3         goal_pos;
13     private float           move_speed;
14
15     private GameObject      target_portal;
16     private Rigidbody       _rigid;
17
18     void OnCollisionEnter(Collision col)...
31     void OnCollisionExit(Collision col)...
35     void Awake()
36     {
37         anim            = GetComponent<Animator>();
38         player_state    = PLAYER_STATE.IDLE;
39         idle_time       = .0f;
40
41         move_speed      = 3.0f;
42         goal_pos        = Vector3.zero;
43
44         target_portal   = null;
45         _rigid          = GetComponent<Rigidbody>();
46     }
47     void Update()...
52     void Update_Inputs()...
89     void Update_Actions()...
142    public void     Add_Pos(Vector3 p)  { transform.position += p; }
143    public void     Set_Pos(Vector3 p)  { transform.position = p; }
144    public Vector3  Get_Pos()           { return transform.position; }
145 }
```

Constants.cs에 정의된 ENTITY_STATE를
사용할 것이기 때문에 삭제

player_state는 _state로
PLAYER_STATE는 ENTITY_STATE로 변경

Player Script 전체를 수정해야 함.

Entity Class 정의

- Entity Script 생성 및 작성
- Player Script 수정

Entity Class 정의

Entity Script 생성 및 작성

- 월드 내 모든 엔티티가 사용하는 기반 스크립트 생성 및 작성
  - Project View - 마우스 오른쪽 클릭 - Create - C# Script
  - 생성한 스크립트의 이름을 "Entity"로 변경

```
1  using UnityEngine;
2  using System.Collections;
3
4  [RequireComponent(typeof(Rigidbody))]
5  public abstract class Entity : MonoBehaviour
6  {
7      [Header("Target")]
8      public Entity _target;
9
10     [Header("ID/Class/Level/State")]
11     [SerializeField] protected string _id;
12     public string ID { get { return _id; } }
13
14     [SerializeField] protected ENTITY_CLASS _class;
15     public ENTITY_CLASS Class { get { return _class; } }
16
17     [SerializeField] protected int _lv;
18     public int Level { get { return _lv; } }
19     public int LevelMax { private set; get; }
20
21     [SerializeField] protected ENTITY_STATE _state;
22     public ENTITY_STATE State { get { return _state; } }
```

Rigidbody가 없는 GameObject에 Entity 클래스를 컴포넌트로 추가하면 Rigidbody도 자동으로 추가됨

Entity 클래스에 의해 Rigidbody는 지울 수 없는 컴포넌트로 보호됨

Inspector View의 필드에 헤더를 추가

Entity Class 정의

- 월드 내 모든 엔티티가 사용하는 기반 스크립트 생성 및 작성(계속)

```
24      [Header("Attributes")]
25      [SerializeField] protected int _strength;
26      public int Strength
27      {
28          get { return _strength; }
29          set { _strength = Mathf.Clamp(value, 0, value); }
30      }
31      [SerializeField] protected int _intelligence;
32      public int Intelligence
33      {
34          get { return _intelligence; }
35          set { _intelligence = Mathf.Clamp(value, 0, value); }
36      }
37      [SerializeField] protected int _health;
38      public int Health
39      {
40          get { return _health; }
41          set { _health = Mathf.Clamp(value, 0, value); }
42      }
43      [SerializeField] protected int _mana;
44      public int Mana
45      {
46          get { return _mana; }
47          set { _mana = Mathf.Clamp(value, 0, value); }
48      }
```

Entity Class 정의

- 월드 내 모든 엔티티가 사용하는 기반 스크립트 생성 및 작성(계속)

```
50      [Header("HP/MP")]
51      [SerializeField] protected bool invincible  = false;
52      [SerializeField] protected int  _hp         = 100;
53      public int HP
54      {
55          get { return _hp; }
56          set { _hp = Mathf.Clamp(value, 0, HPMax); }
57      }
58      public abstract int HPRecovery  { get; }
59      public abstract int HPMax       { get; }
60
61      [SerializeField] protected int  _mp = 100;
62      public int MP
63      {
64          get { return _mp; }
65          set { _mp = Mathf.Clamp(value, 0, MPMax); }
66      }
67      public abstract int MPRecovery  { get; }
68      public abstract int MPMax       { get; }
```

Entity Class 정의

- 월드 내 모든 엔티티가 사용하는 기반 스크립트 생성 및 작성(계속)

```
70      public abstract int Physics_Damage  { get; }
71      public abstract int Magic_Damage    { get; }
72      public abstract int Physics_Defense { get; }
73      public abstract int Magic_Defense   { get; }
74
75      public float move_speed { protected set; get; }
76
77      /// <summary>
78      /// Entity의 HP/MP 회복 함수
79      /// </summary>
80      protected virtual void Recovery()
81      {
82          if ( !enabled ) return;
83
84          if ( _hp < HPMax )
85          {
86              _hp += HPRecovery;
87              if ( _hp >= HPMax ) _hp = HPMax;
88          }
89          if ( _mp < MPMax )
90          {
91              _mp += MPRecovery;
92              if ( _mp >= MPMax ) _mp = MPMax;
93          }
94      }
95      public float Hp_Percent() { return (_hp != 0 && HPMax != 0) ? (float)_hp/(float)HPMax : 0.0f; }
96      public float Mp_Percent() { return (_mp != 0 && MPMax != 0) ? (float)_mp/(float)MPMax : 0.0f; }
97
98      public void Init()
99      {
100         LevelMax = 99;
101
102         HP = HPMax;
103         MP = MPMax;
104
105         InvokeRepeating("Recovery", 1.0f, 1.0f);
106     }
107 }
```

특정 시간에 한번씩 반복 호출할 때 사용하는 유니티 함수
InvokeRepeating(호출할 함수명, 최초 실행 지연 시간, 반복 실행 시 지연 시간)

Entity Class 정의

- **Player Script 수정**
  - Player Script 수정

```
1  using UnityEngine;
2  using System.Collections;
3
4  [RequireComponent(typeof(Animator))]
5  public class Player : Entity
6  {
7      protected override void Recovery()
8      {
9          base.Recovery();
10
11         int buff_bonus_hp = 0;
12         HP += buff_bonus_hp;
13
14         int buff_bonus_mp = 0;
15         MP += buff_bonus_mp;
16     }
```

Entity Class 정의

- Player Script 수정(계속)

```
17  public override int HPMax
18  {
19      get
20      {
21          int base_hp = 100 + Level * 10;
22          int equip_bonus = 0;
23          int buff_bonus = 0;
24          int attr_bonus = Health * 20;
25
26          return base_hp + equip_bonus + buff_bonus + attr_bonus;
27      }
28  }
29  public override int HPRecovery
30  {
31      get
32      {
33          int base_HpRec = Level;
34          int equip_bonus = 0;
35          int buff_bonus = 0;
36          int attr_bonus = Health;
37
38          return base_HpRec + equip_bonus + buff_bonus + attr_bonus;
39      }
40  }
```

Entity Class 정의

- Player Script 수정(계속)

```
41  public override int MPMax
42  {
43      get
44      {
45          int base_mp = 100 + Level * 10;
46          int equip_bonus = 0;
47          int buff_bonus = 0;
48          int attr_bonus = Mana * 10;
49
50          return base_mp + equip_bonus + buff_bonus + attr_bonus;
51      }
52  }
53  public override int MPRecovery
54  {
55      get
56      {
57          int base_MpRec = Level;
58          int equip_bonus = 0;
59          int buff_bonus = 0;
60          int attr_bonus = Mana;
61
62          return base_MpRec + equip_bonus + buff_bonus + attr_bonus;
63      }
64  }
```

Entity Class 정의

- Player Script 수정(계속)

```
65  public override int Physics_Damage
66  {
67      get
68      {
69          int base_dmg = 10 + Level;
70          int equip_bonus = 0;
71          int buff_bonus = 0;
72          int attr_bonus = Strength * 2;
73
74          return base_dmg + equip_bonus + buff_bonus + attr_bonus;
75      }
76  }
77  public override int Magic_Damage
78  {
79      get
80      {
81          int base_mdmg = 10 + Level*2;
82          int equip_bonus = 0;
83          int buff_bonus = 0;
84          int attr_bonus = Intelligence * 4;
85
86          return base_mdmg + equip_bonus + buff_bonus + attr_bonus;
87      }
88  }
```

Entity Class 정의

- Player Script 수정(계속)

```
89  public override int Physics_Defense
90  {
91      get
92      {
93          int base_def = 5 + Level;
94          int equip_bonus = 0;
95          int buff_bonus = 0;
96          int attr_bonus = Strength + Health;
97
98          return base_def + equip_bonus + buff_bonus + attr_bonus;
99      }
100 }
101 public override int Magic_Defense
102 {
103     get
104     {
105         int base_mdef = 5 + Level;
106         int equip_bonus = 0;
107         int buff_bonus = 0;
108         int attr_bonus = Intelligence + Mana;
109
110         return base_mdef + equip_bonus + buff_bonus + attr_bonus;
111     }
112 }
```

Entity Class 정의

- Player Script 수정(계속)

```
114     [Header("Attribute Points")]
115     [SerializeField] private int _att_point;
116     public int Att_Point
117     {
118         get { return _att_point; }
119         set { _att_point = Mathf.Clamp(value, 0, value); }
120     }
121
122     [Header("Experience")]
123     [SerializeField] long _exp;
124     public long Exp
125     {
126         get { return _exp; }
127         set
128         {
129             _exp = value;
130             if ( _exp >= ExpMax )
131             {
132                 _exp -= ExpMax;
133
134                 if ( Level*0.3 < 1.0 )  Att_Point ++;
135                 else                    Att_Point = Att_Point + (int)(Level*0.3);
136
137                 _lv ++;
138
139                 HP = HPMax;
140                 MP = MPMax;
141             }
142         }
143     }
144     public float Exp_Percent()  { return (Exp!=0 && ExpMax!=0) ? (float)Exp/(float)ExpMax : 0.0f; }
145     public long  ExpMax         { get { return Level*Level*100; } }
```

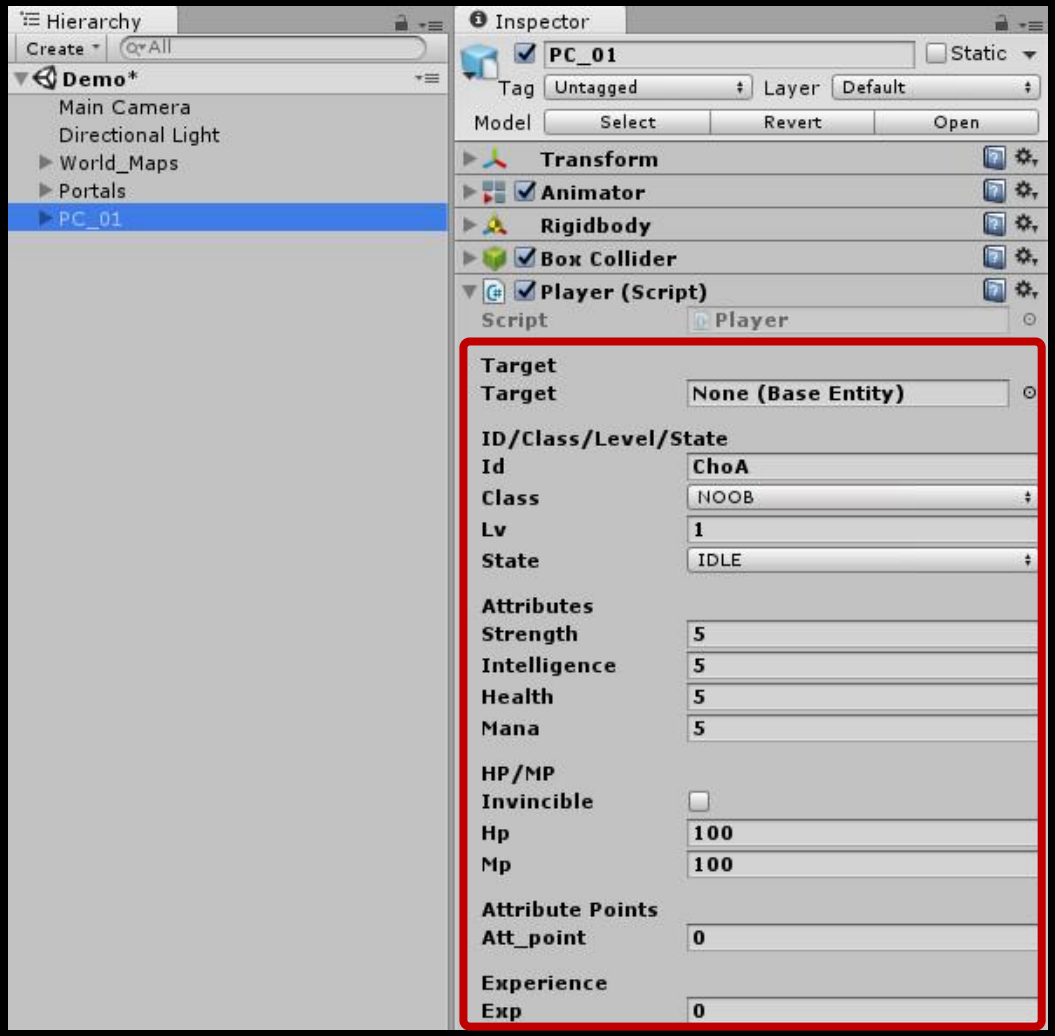
Entity Class 정의

- Player Script 수정(계속)

```
147     private Animator        anim;
148     private float           idle_time;
149
150     private Vector3         goal_pos;
151     private float           move_speed;
152
153     private GameObject      target_portal;
154     private Rigidbody       _rigid;
155
156     void OnCollisionEnter(Collision col)...
169     void OnCollisionExit(Collision col)...
173     void Awake()
174     {
175         base.Init();
176
177         anim            = GetComponent<Animator>();
178         _state          = ENTITY_STATE.IDLE;
179         idle_time       = .0f;
180
181         move_speed      = 3.0f;
182         goal_pos        = Vector3.zero;
183
184         target_portal   = null;
185         _rigid          = GetComponent<Rigidbody>();
186     }
187     void Update()...
192     void Update_Inputs()...
229     void Update_Actions()...
282     public void     Add_Pos(Vector3 p)  { transform.position += p; }
283     public void     Set_Pos(Vector3 p)  { transform.position = p; }
284     public Vector3  Get_Pos()           { return transform.position; }
285 }
```

Entity Class 정의

- Player Script 변수 설정
  - 그림과 같이 변수들의 값을 설정

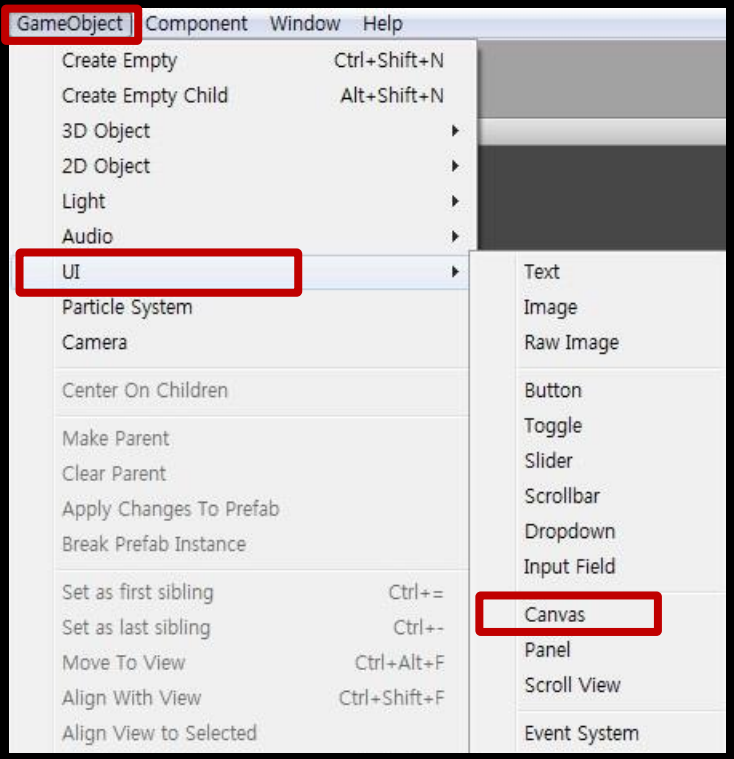


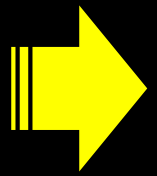
Base UI 설정

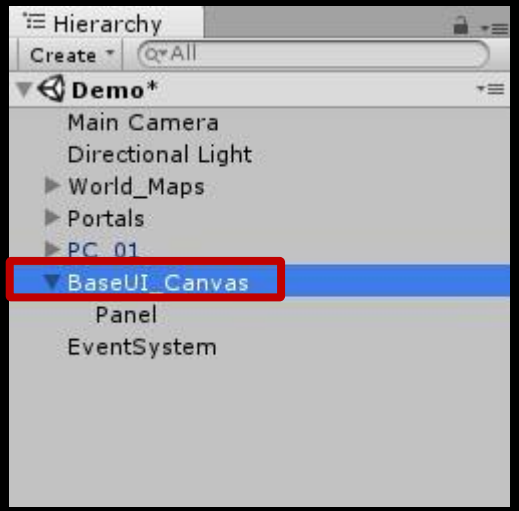
- HP/MP
- Experience
- Shortcuts

Base UI 설정

- **HP/MP**
  - Base UI를 배치할 Canvas 생성
    - GameObject - UI - Canvas
    - 생성한 Canvas의 이름을 "BaseUI_Canvas"로 변경

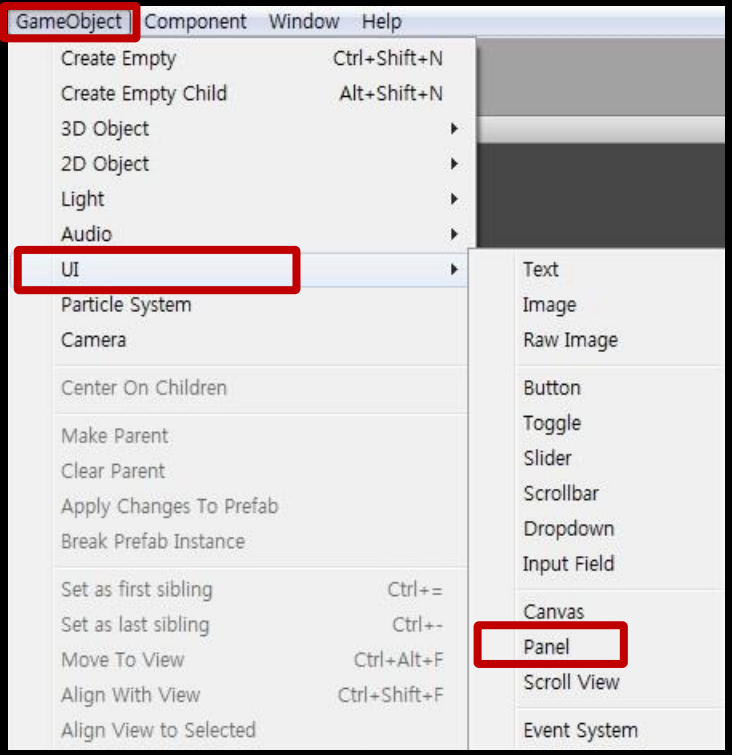


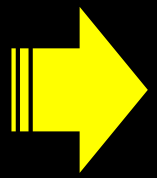


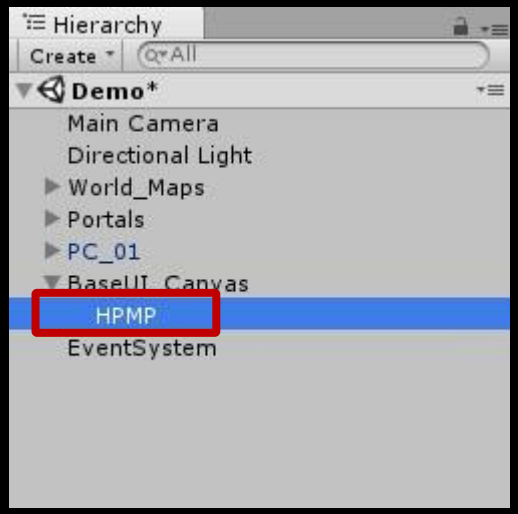


Base UI 설정

- HP/MP UI를 관리할 Panel 생성 및 설정
  - GameObject - UI - Panel
  - 생성한 Panel의 이름을 "HPMP"로 변경

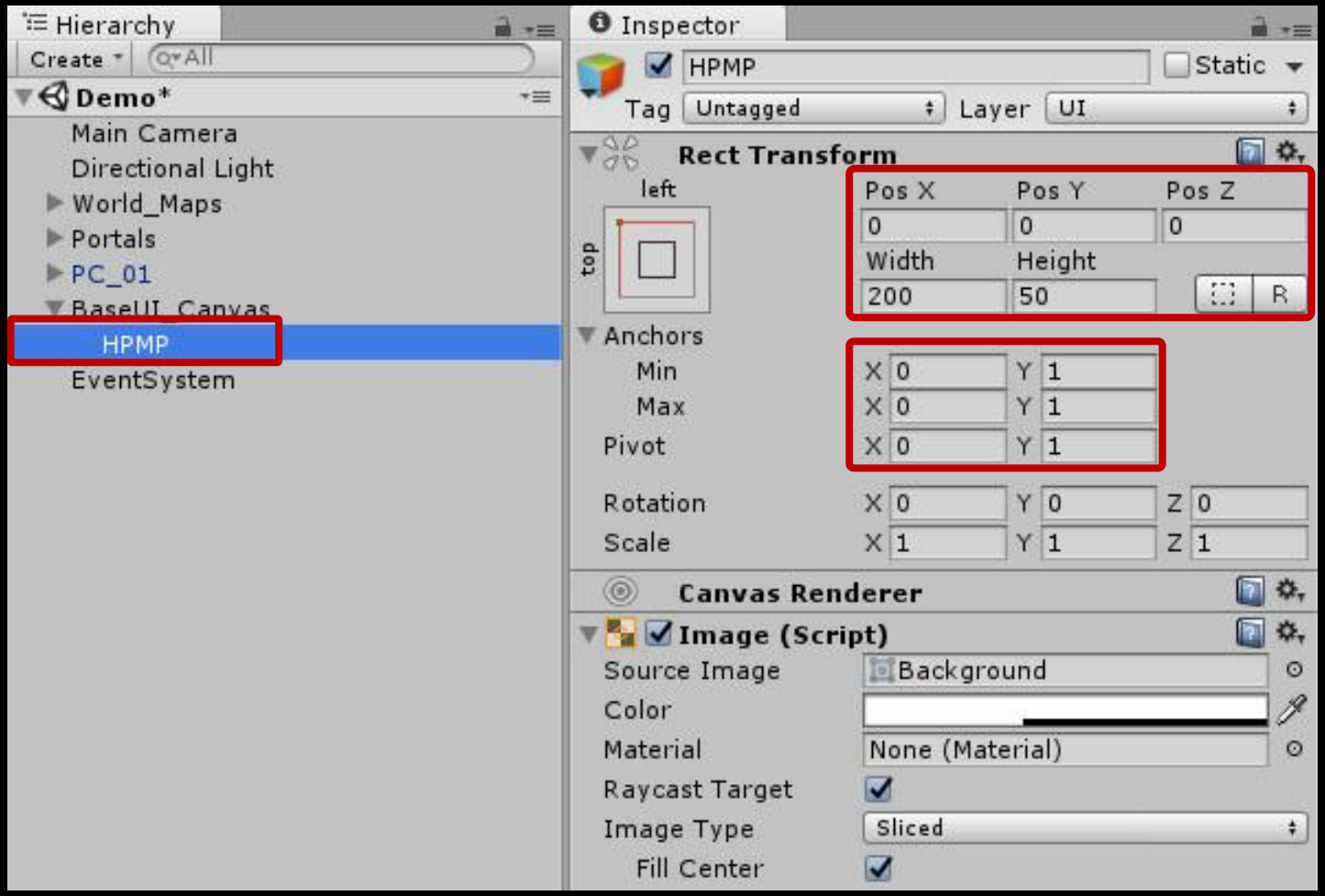






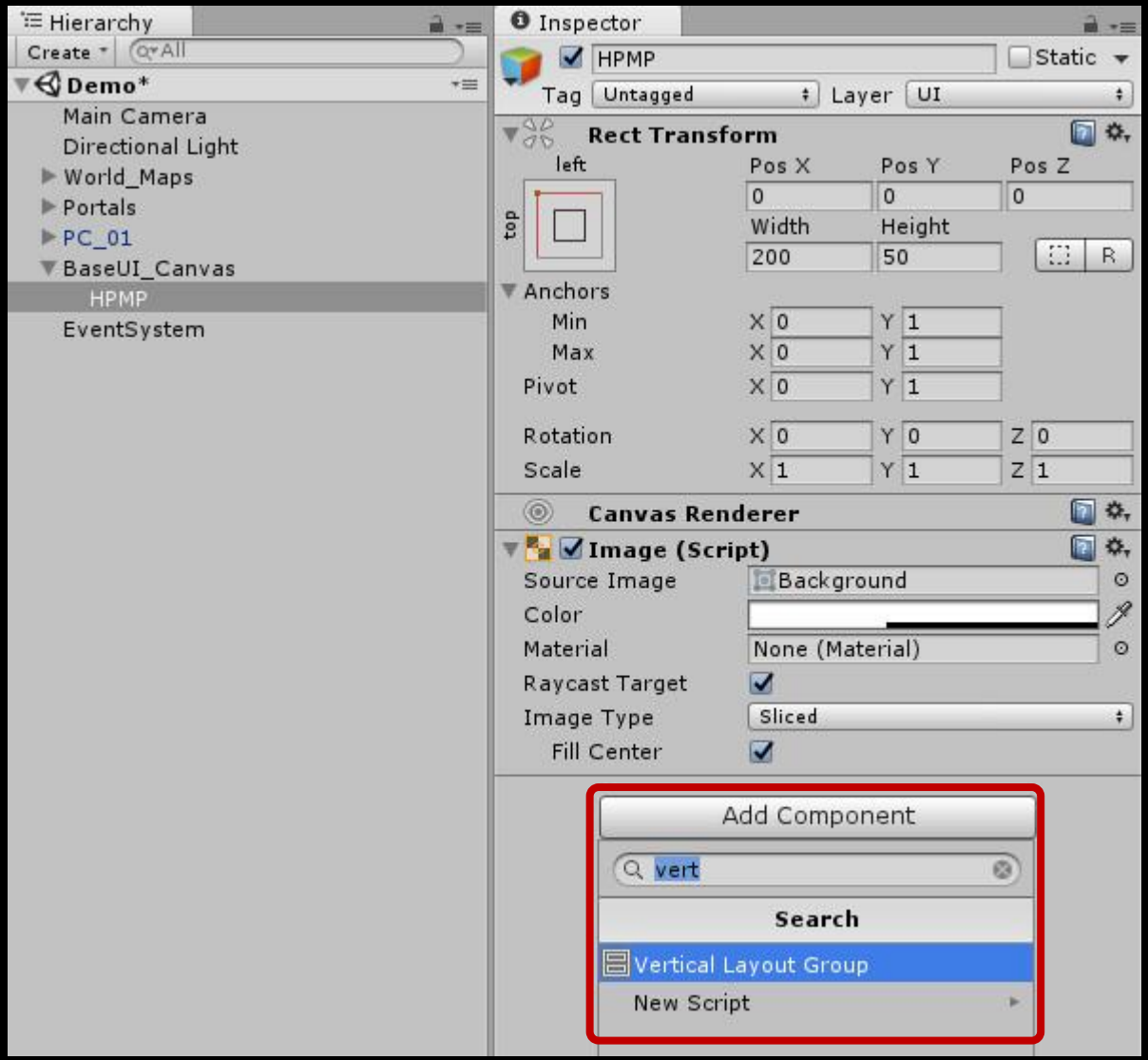
Base UI 설정

- HP/MP UI를 관리할 Panel 생성 및 설정(계속)
  - Anchors 값과 Pivot값을 변경한 후 위치와 크기 값을 변경



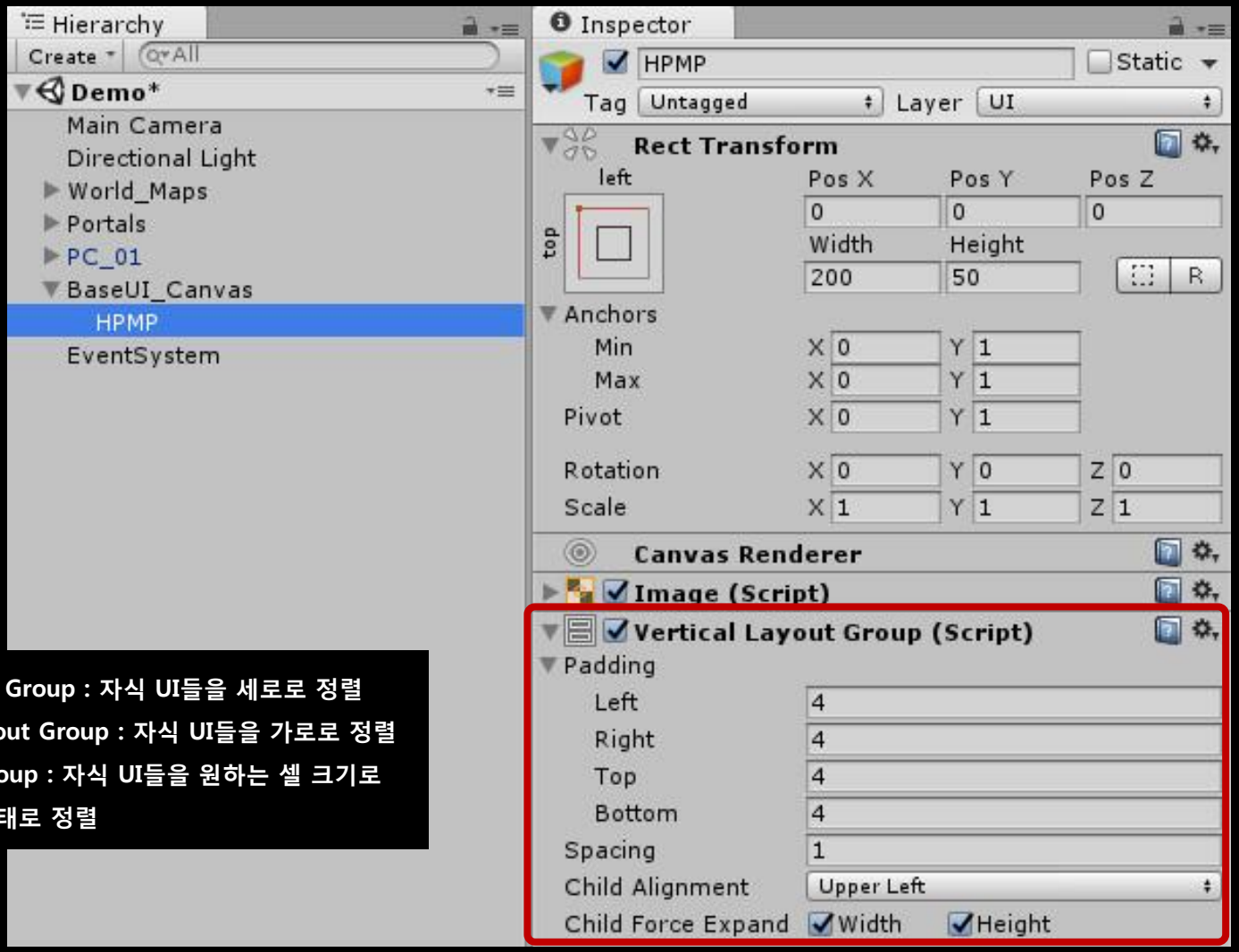
Base UI 설정

- UI 정렬을 제공하는 "Vertical Layout Group" 컴포넌트 추가 및 설정(계속)



Base UI 설정

- UI 정렬을 제공하는 "Vertical Layout Group" 컴포넌트 추가 및 설정(계속)



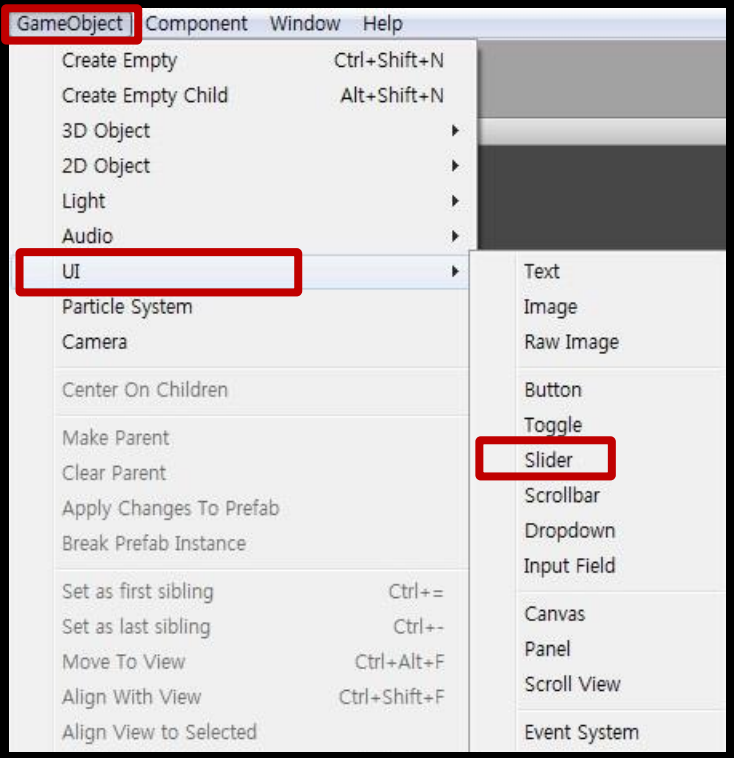
Vertical Layout Group : 자식 UI들을 세로로 정렬

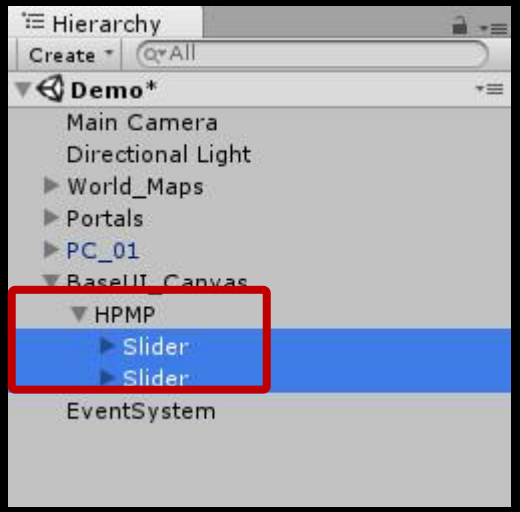
Horizontal Layout Group : 자식 UI들을 가로로 정렬

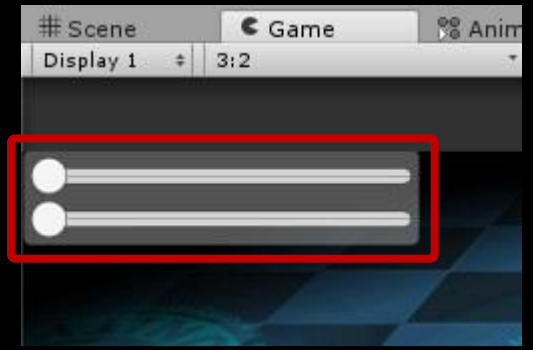
Grid Layout Group : 자식 UI들을 원하는 셀 크기로 격자 셀 구조 형태로 정렬

Base UI 설정

- 체력(HP)을 제어할 Slider 생성 및 설정
  - GameObject - UI - Slider
  - 생성한 Slider를 "HPMP" Panel의 자식으로 적용

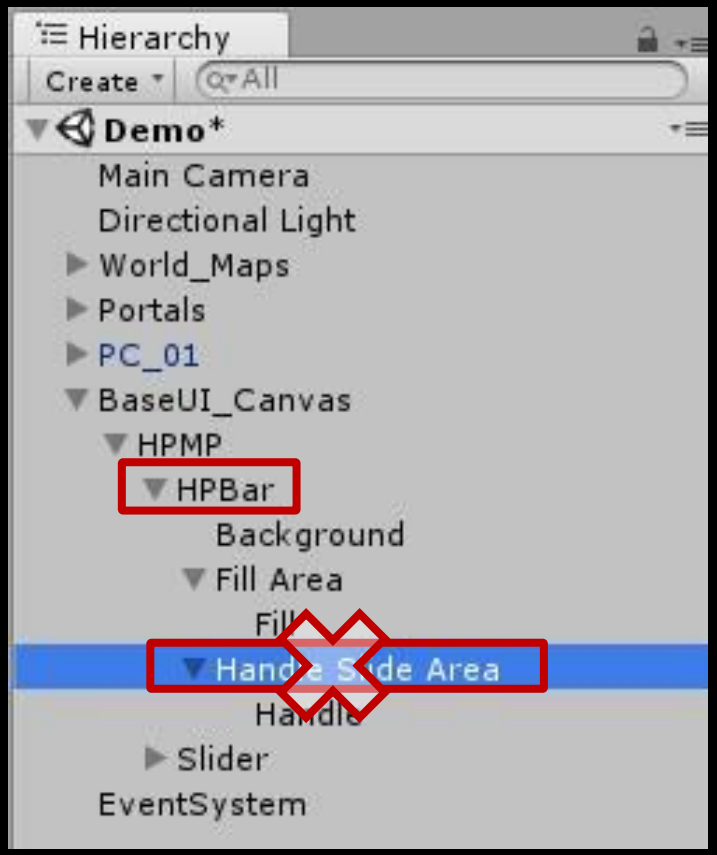






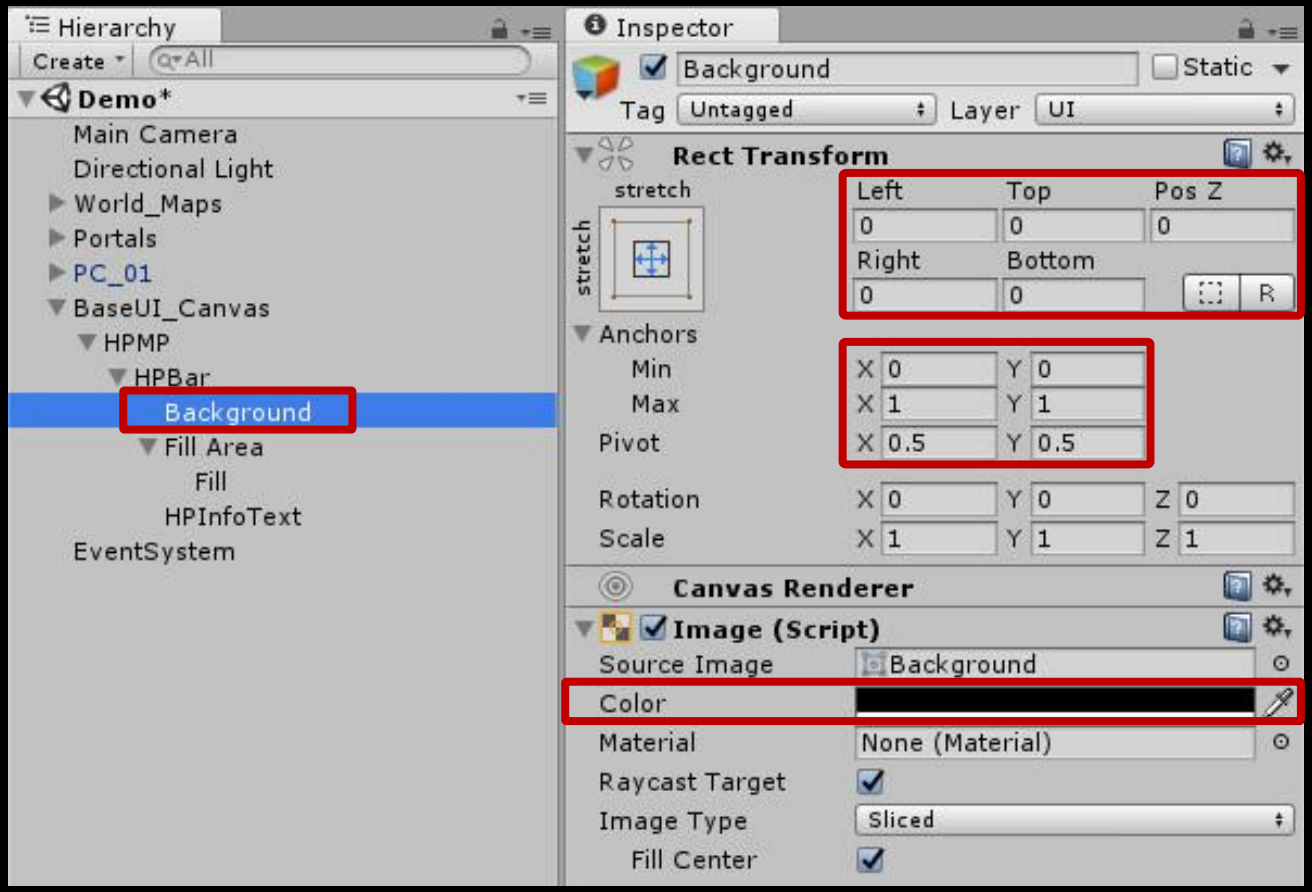
Base UI 설정

- 체력(HP)을 제어할 Slider 생성 및 설정(계속)
  - Slider의 이름을 "HPBar"로 변경
  - "Handle Slide Area"는 사용하지 않으므로 삭제



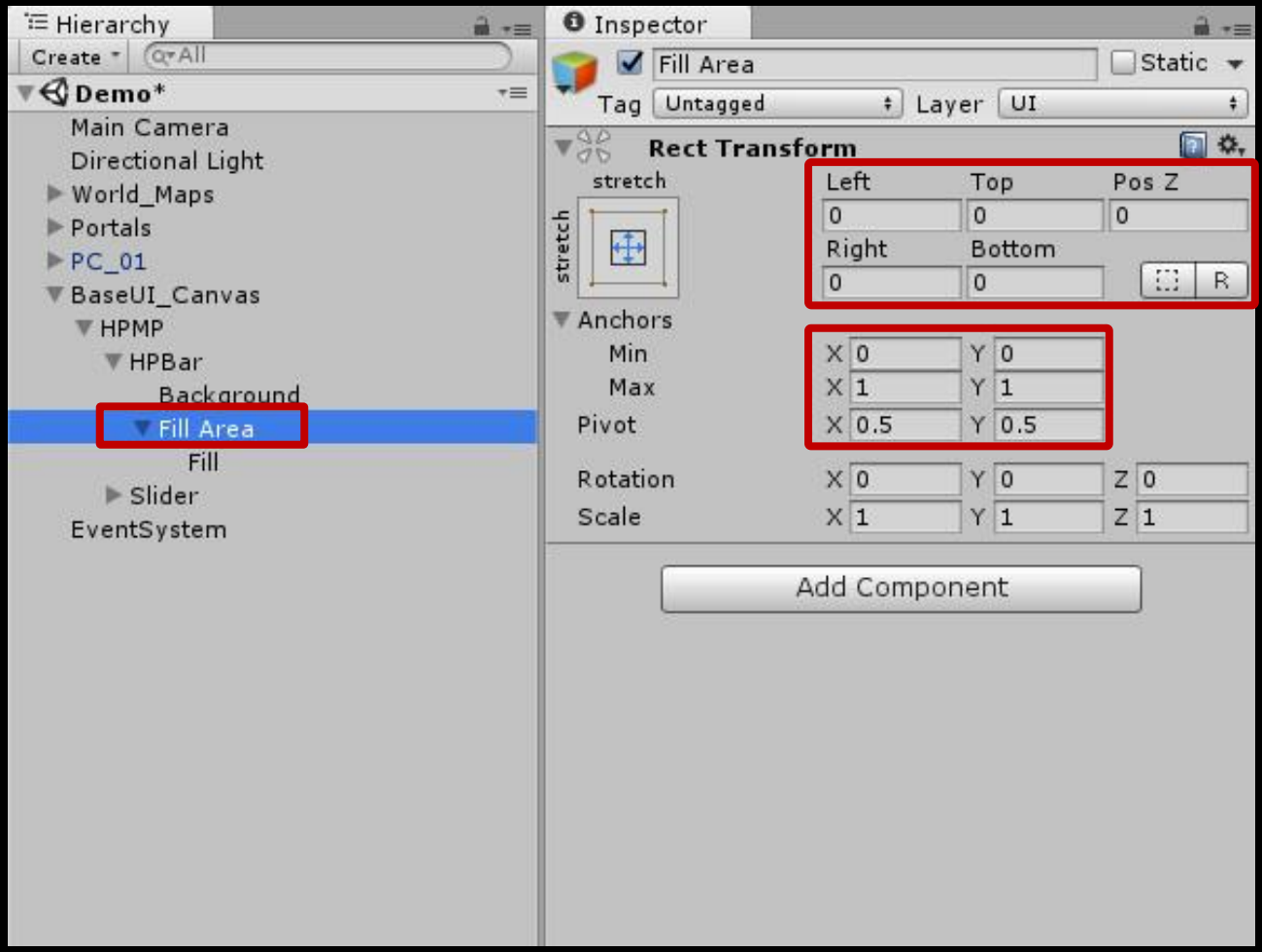
Base UI 설정

- 체력(HP)을 제어할 Slider 생성 및 설정(계속)
  - "Background"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 검은색으로 변경



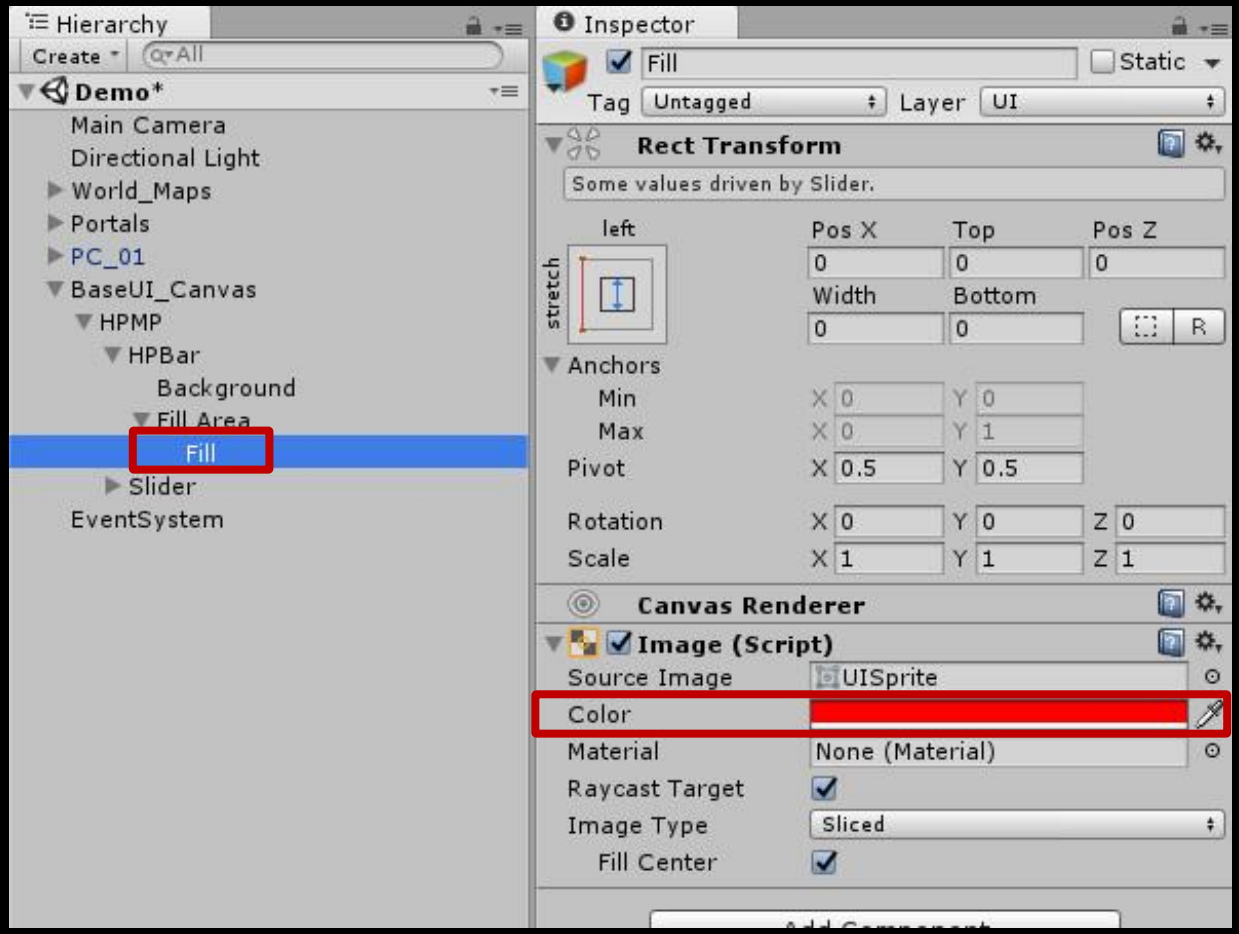
Base UI 설정

- 체력(HP)을 제어할 Slider 생성 및 설정(계속)
  - "Fill Area"의 Rect Transform 컴포넌트 설정



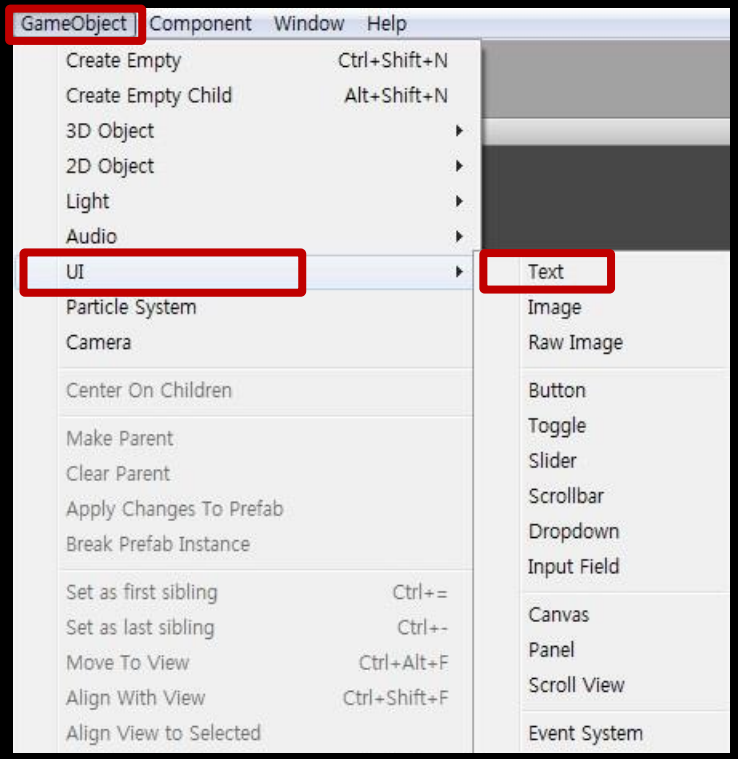
Base UI 설정

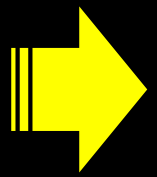
- 체력(HP)을 제어할 Slider 생성 및 설정(계속)
  - "Fill"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 빨간색으로 변경

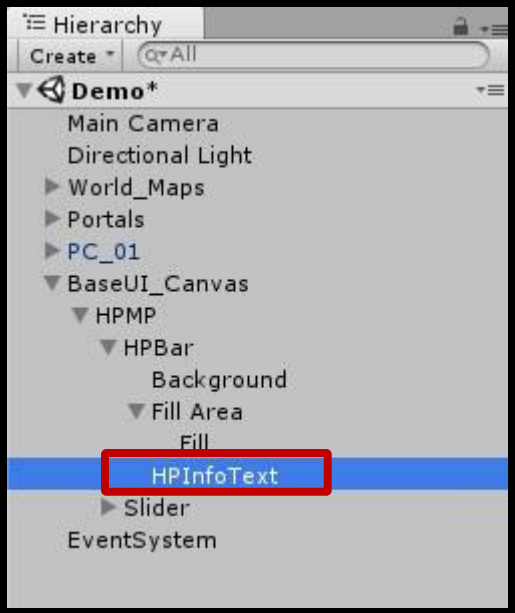


Base UI 설정

- 체력 데이터를 표현할 Text 생성 및 설정
  - GameObject - UI - Text
  - 생성한 Text의 위치를 "HPBar" Slider의 자식으로 변경
  - Text의 이름을 "HPInfoText"로 변경

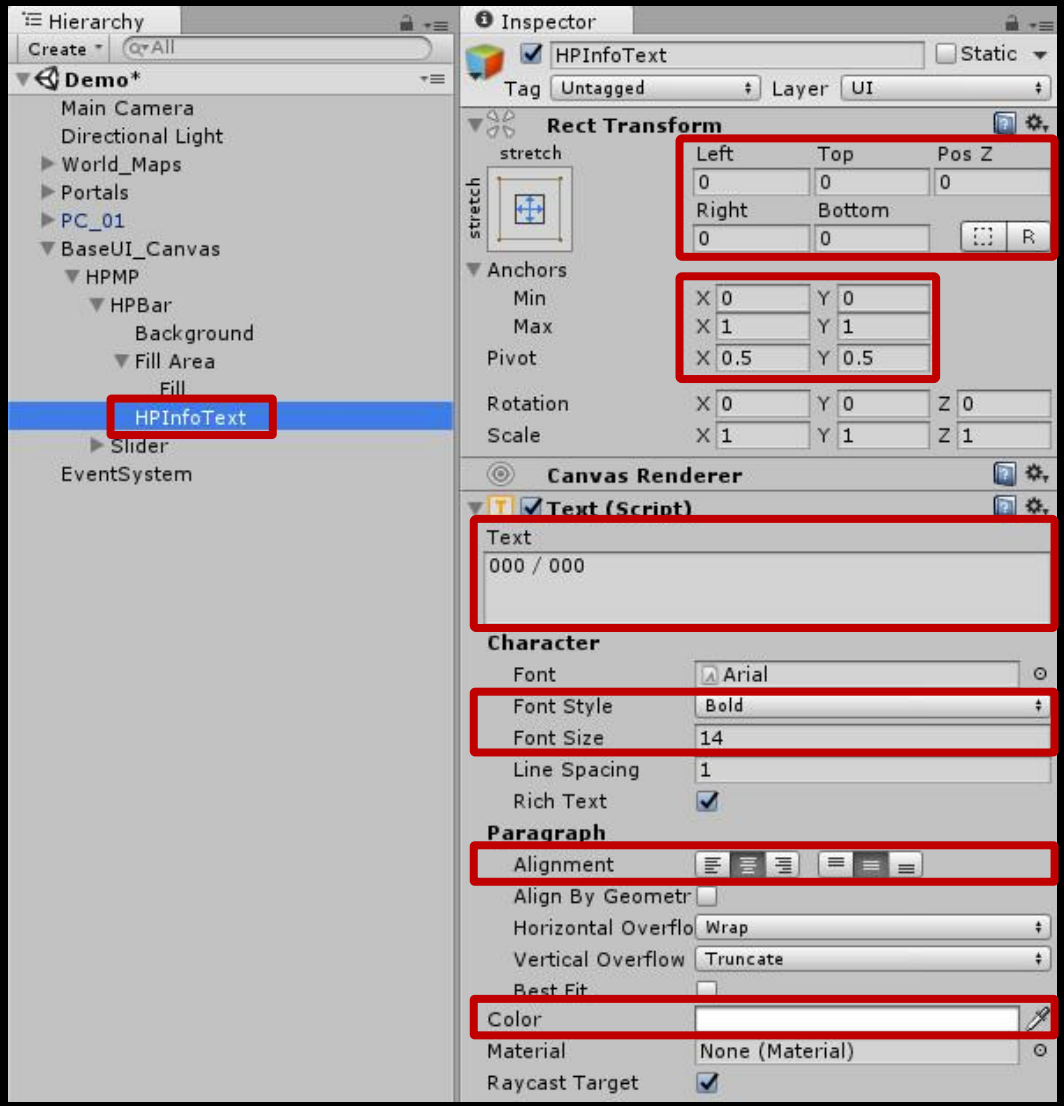






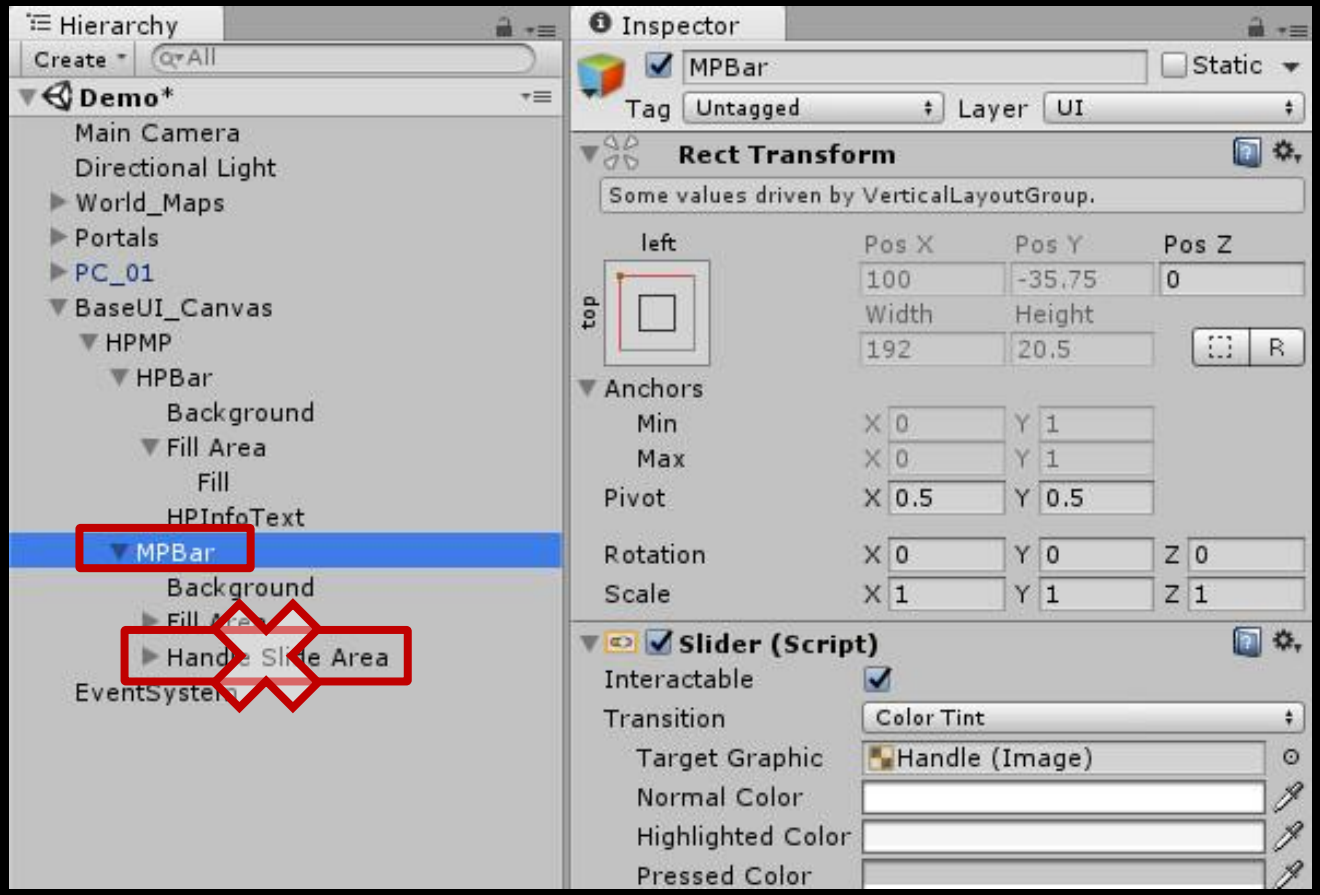
Base UI 설정

- 체력 데이터를 표현할 Text 생성 및 설정(계속)



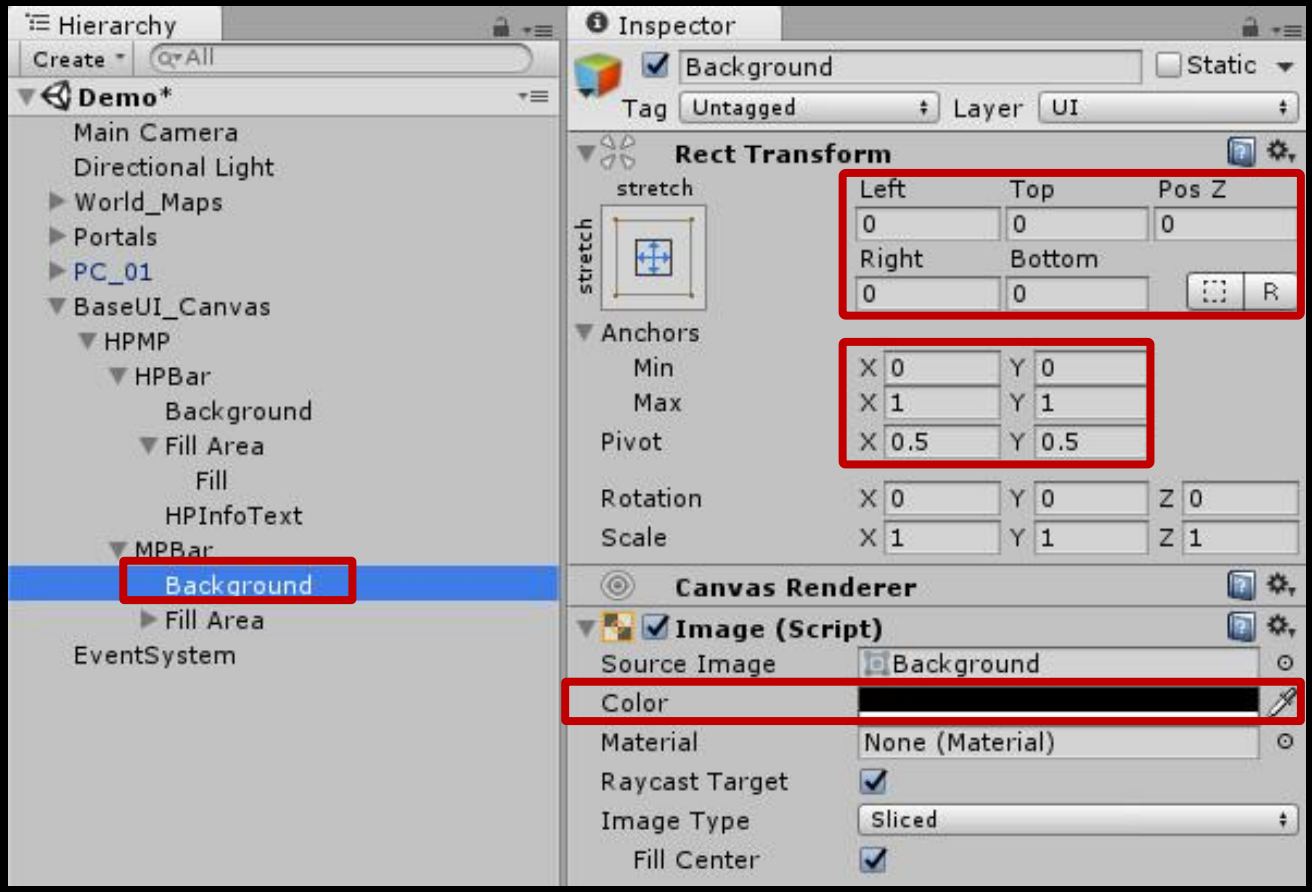
Base UI 설정

- 마나(MP)를 제어할 Slider 설정
  - 생성해 두었던 Slider의 이름을 "MPBar"로 변경
  - "Handle Slide Area"는 사용하지 않으므로 삭제



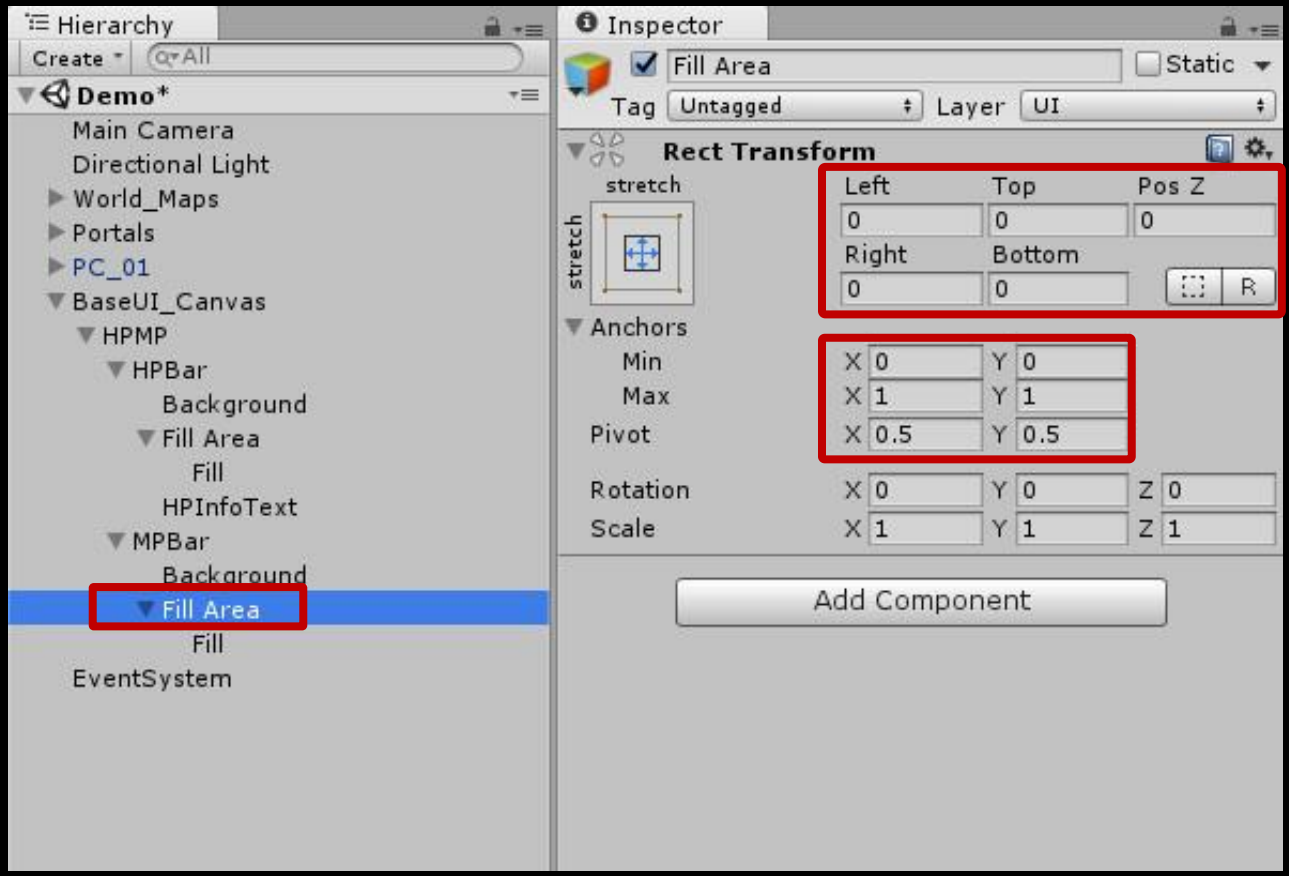
Base UI 설정

- 마나(MP)를 제어할 Slider 설정(계속)
  - "Background"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 검은색으로 변경



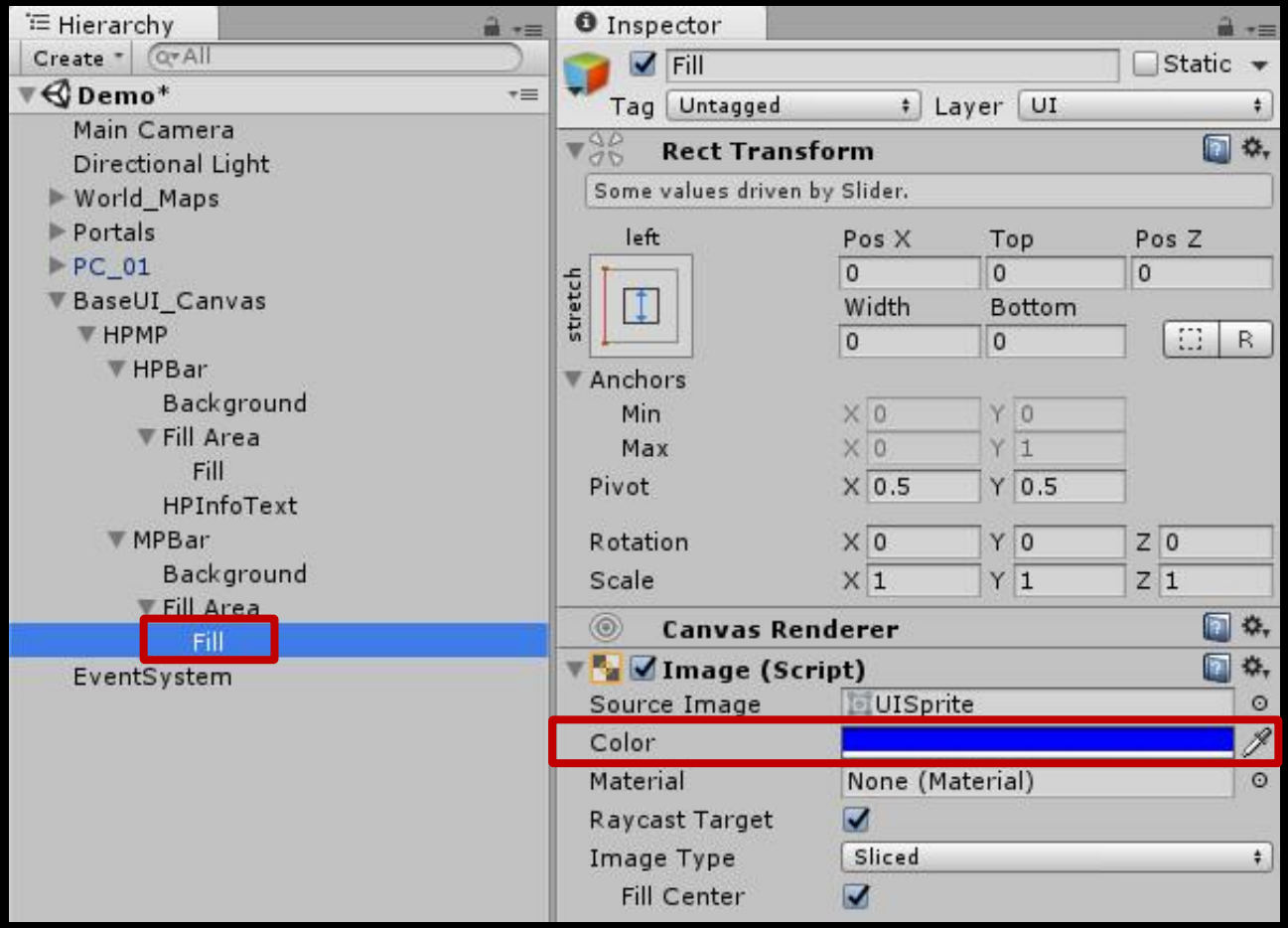
Base UI 설정

- 마나(MP)를 제어할 Slider 설정(계속)
  - "Fill Area"의 Rect Transform 컴포넌트 설정



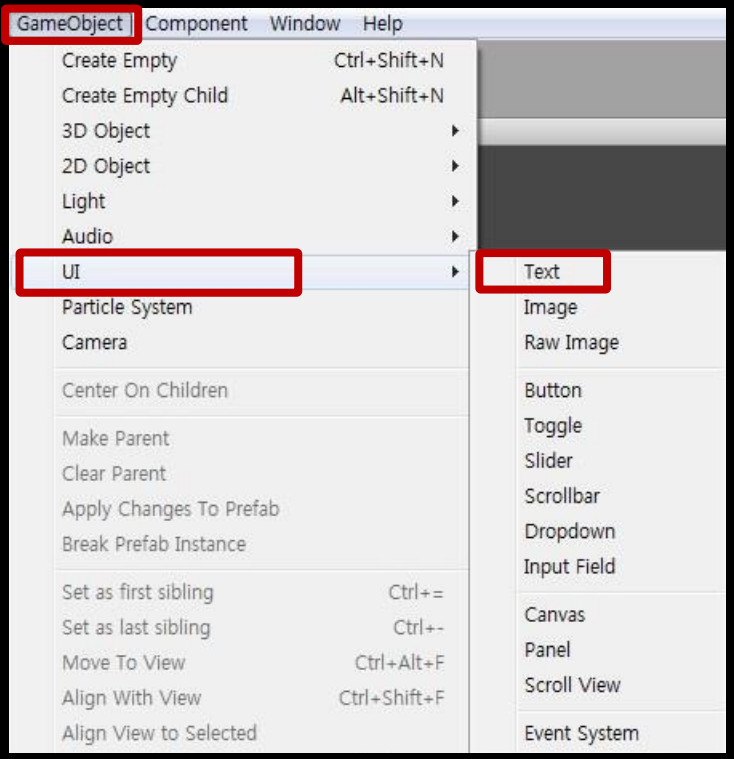
Base UI 설정

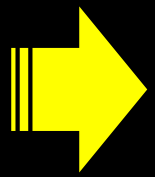
- 마나(MP)를 제어할 Slider 설정(계속)
  - "Fill"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 빨간색으로 변경

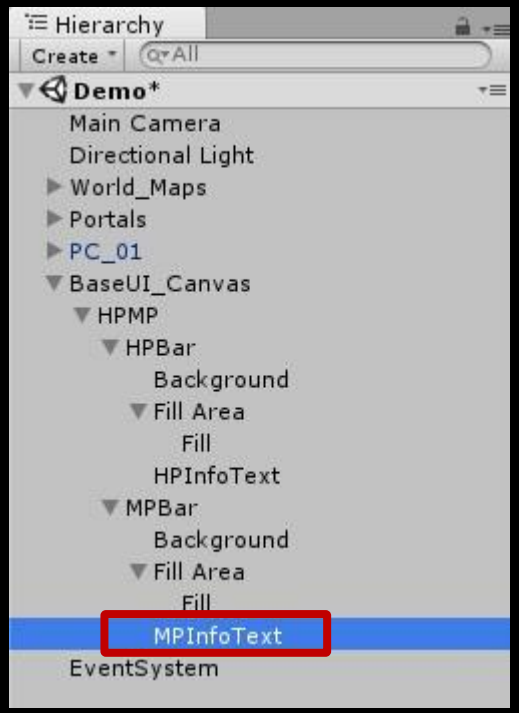


Base UI 설정

- 마나 데이터를 표현할 Text 생성 및 설정
  - GameObject - UI - Text
  - 생성한 Text의 위치를 "MPBar" Slider의 자식으로 변경
  - Text의 이름을 "MPInfoText"로 변경

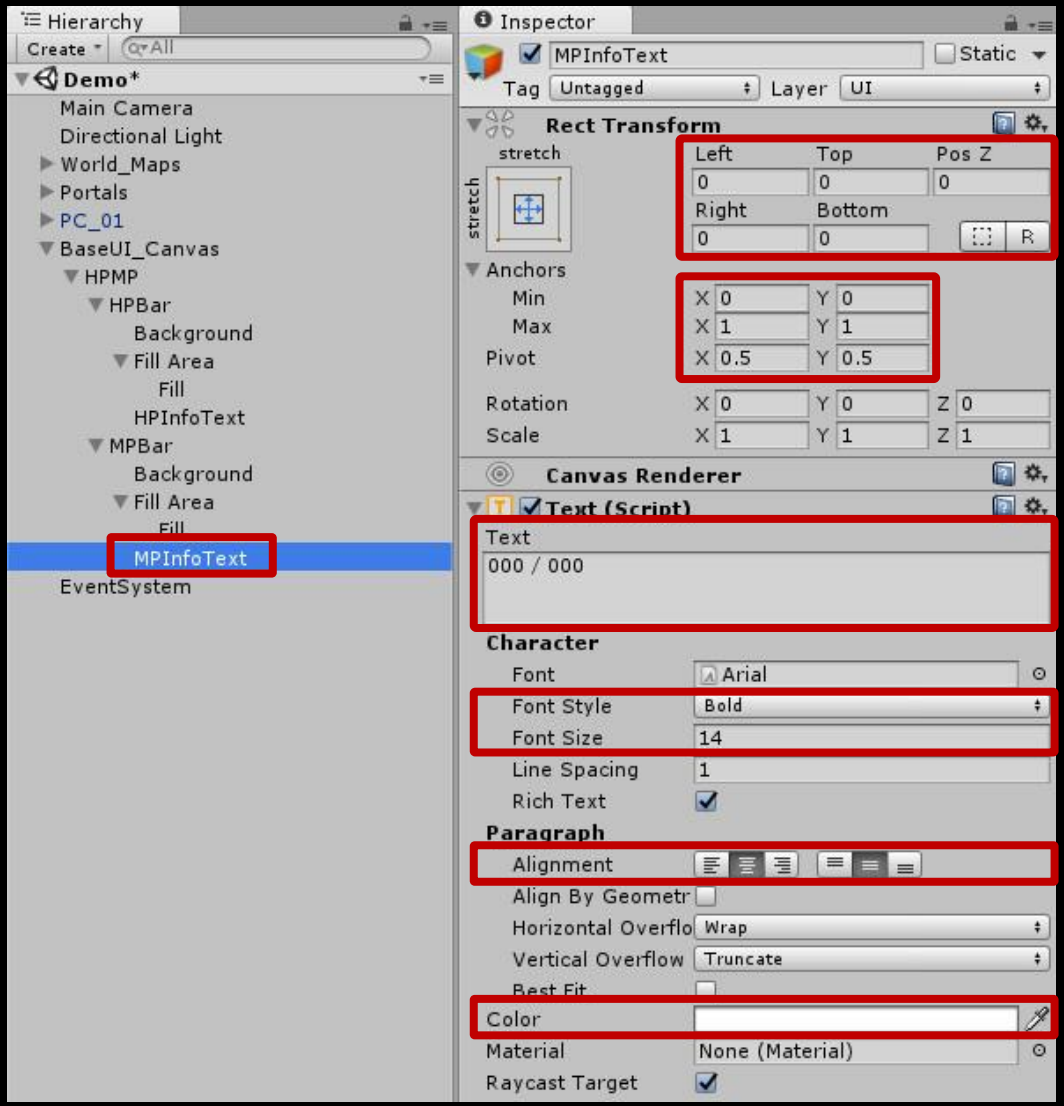






Base UI 설정

- 마나 데이터를 표현할 Text 생성 및 설정(계속)



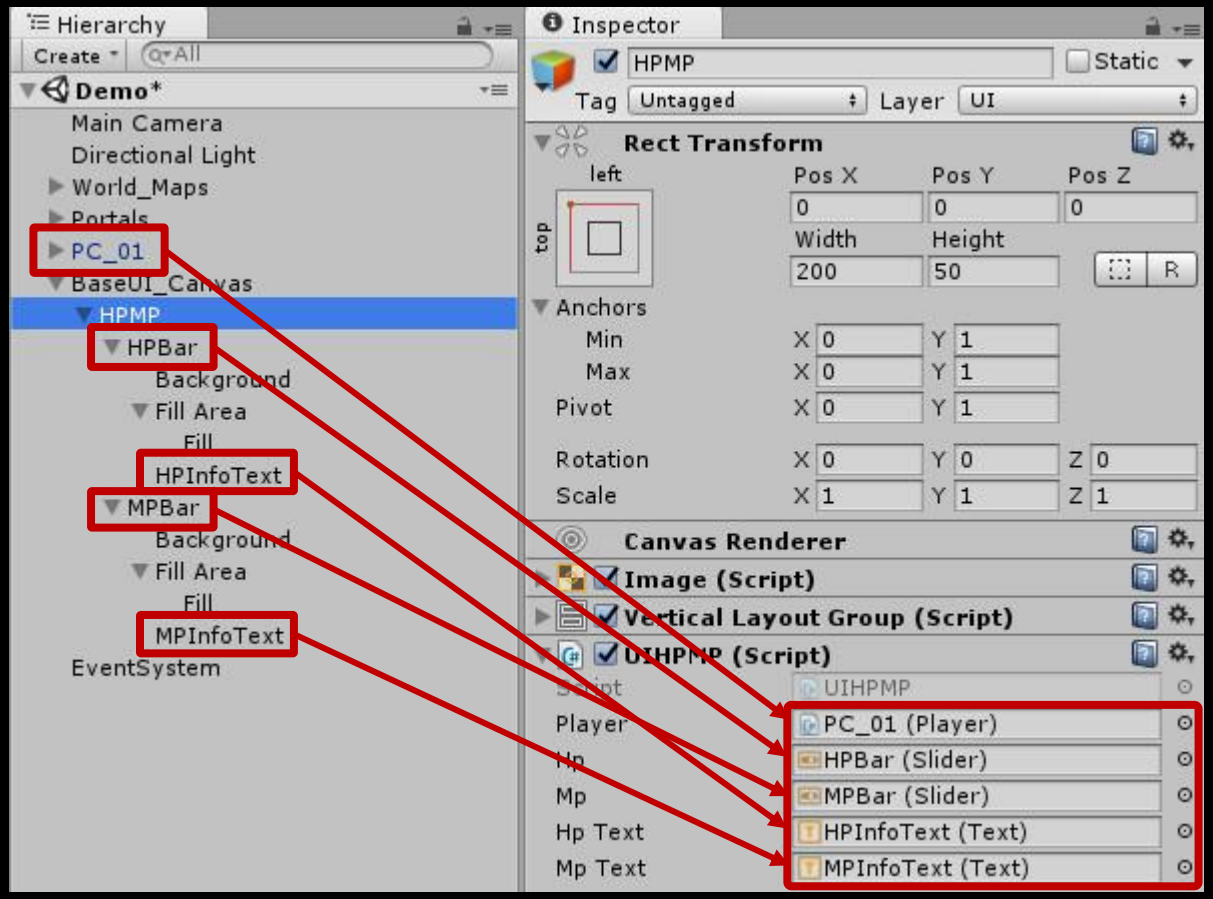
Base UI 설정

- 데이터와 UI 연결을 위한 스크립트 생성 및 작성
  - Project View - 마우스 오른쪽 클릭 - Create - C# Script
  - 생성한 스크립트의 이름을 "UIHPMP"로 변경
  - 스크립트를 "HPMP" Panel의 컴포넌트로 적용

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class UIHPMP : MonoBehaviour
6  {
7      [SerializeField] Player _player;
8      [SerializeField] Slider _hp;
9      [SerializeField] Slider _mp;
10     [SerializeField] Text   _hpText;
11     [SerializeField] Text   _mpText;
12
13     void Update()
14     {
15         if ( _player.transform == null ) return;
16
17         _hp.value    = _player.Hp_Percent();
18         _hpText.text = _player.HP + "/" + _player.HPMax;
19         _mp.value    = _player.Mp_Percent();
20         _mpText.text = _player.MP + "/" + _player.MPMax;
21     }
22 }
```

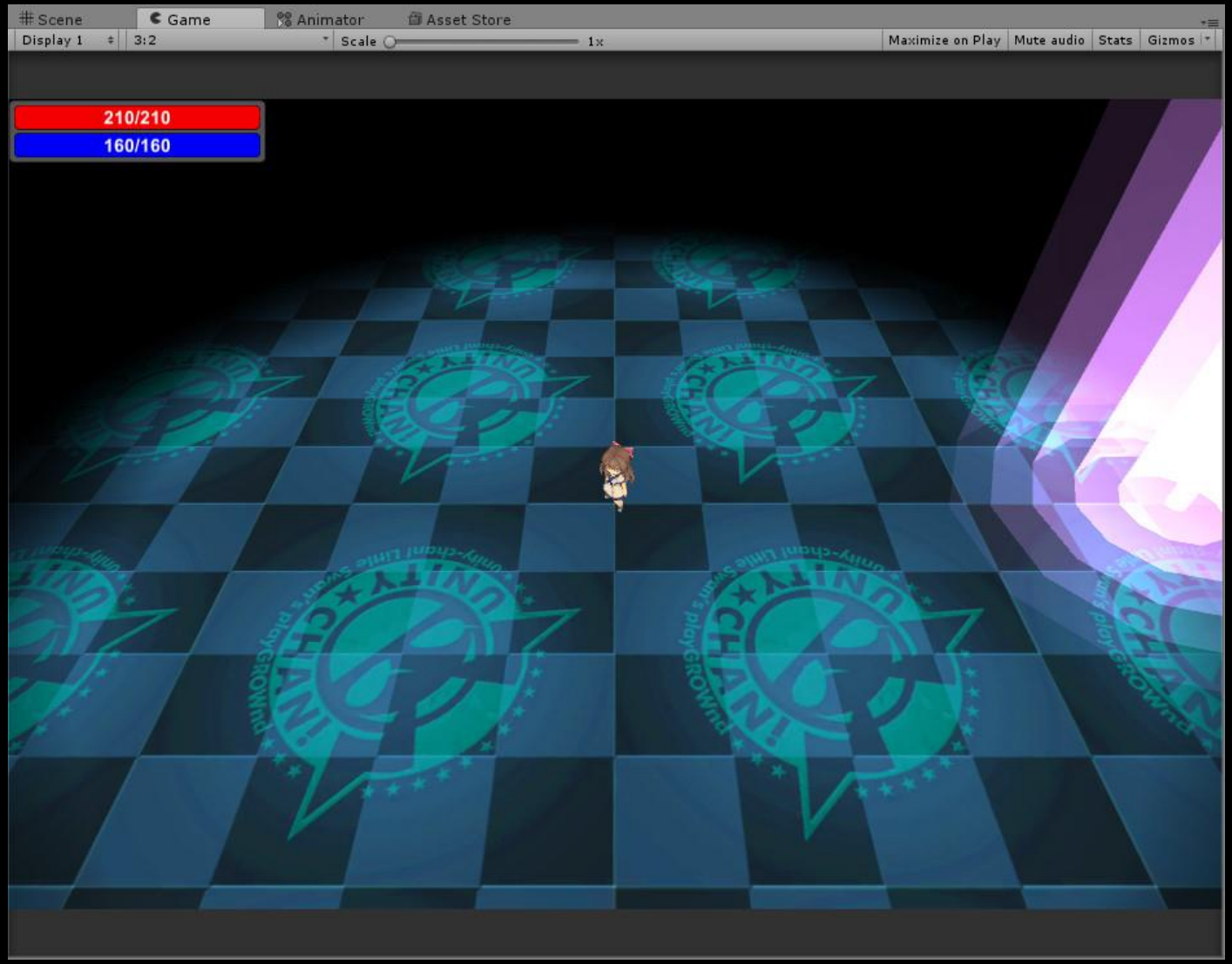
Base UI 설정

- UIHPMP Script 변수 설정
  - Player 변수에 HP, MP 정보가 들어있는 캐릭터 클래스 저장
  - 정보를 변경할 Slider와 Text를 저장



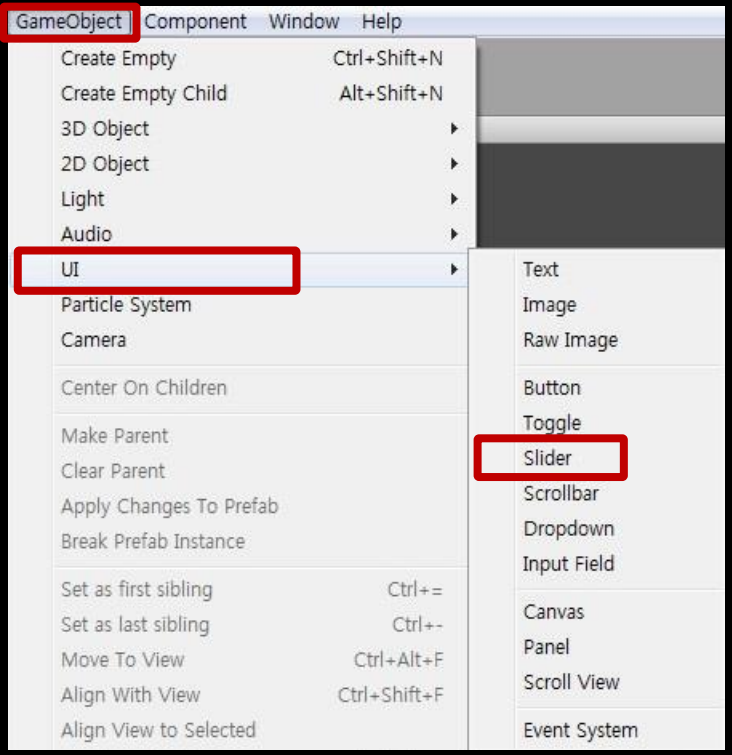
Base UI 설정

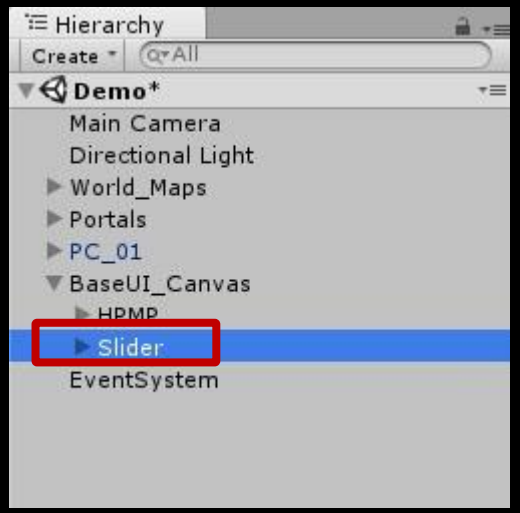
- 결과 화면



Base UI 설정

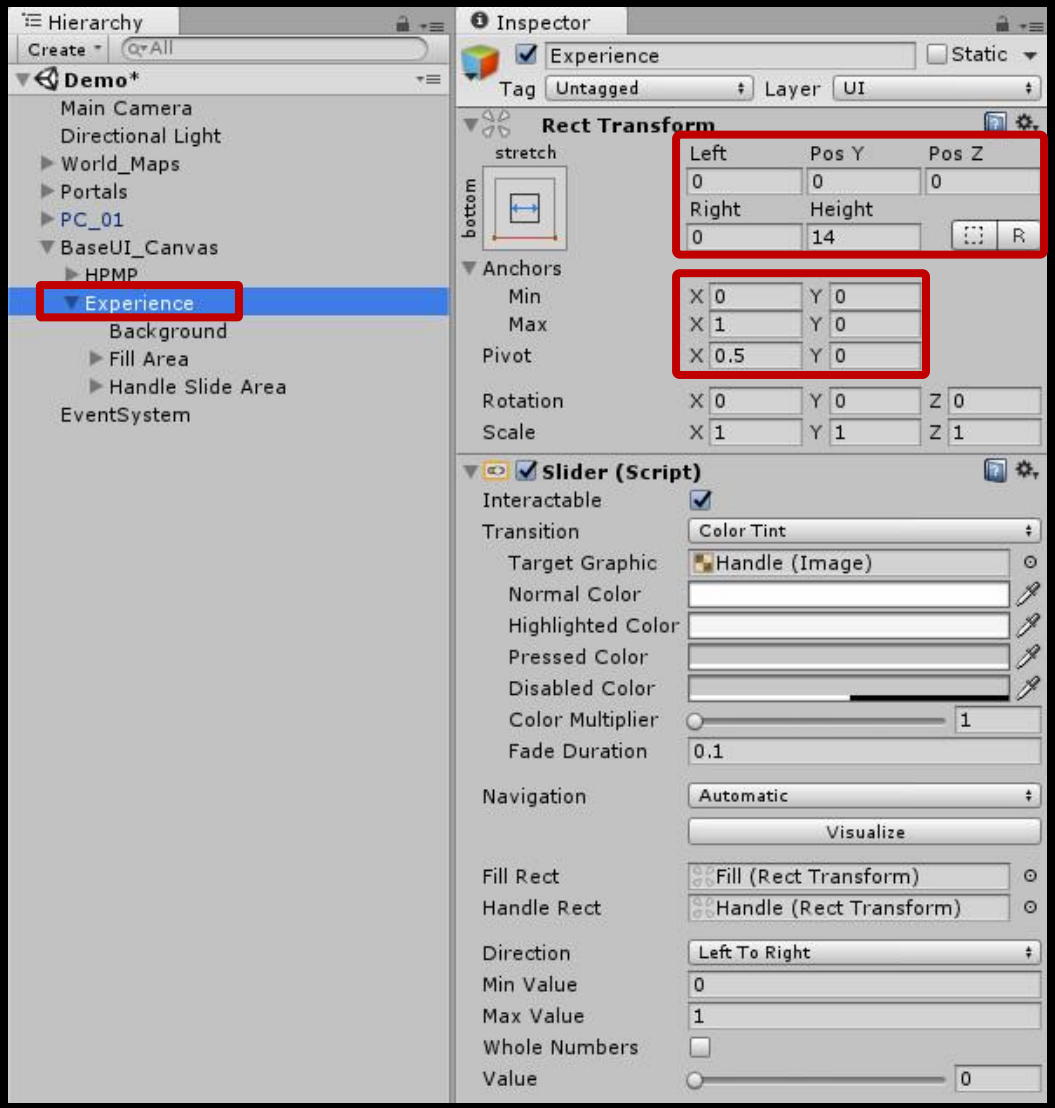
- Experience를 관리할 Slider 생성 및 설정
  - GameObject - UI - Slider
  - 생성한 Slider의 이름을 "Experience"로 변경





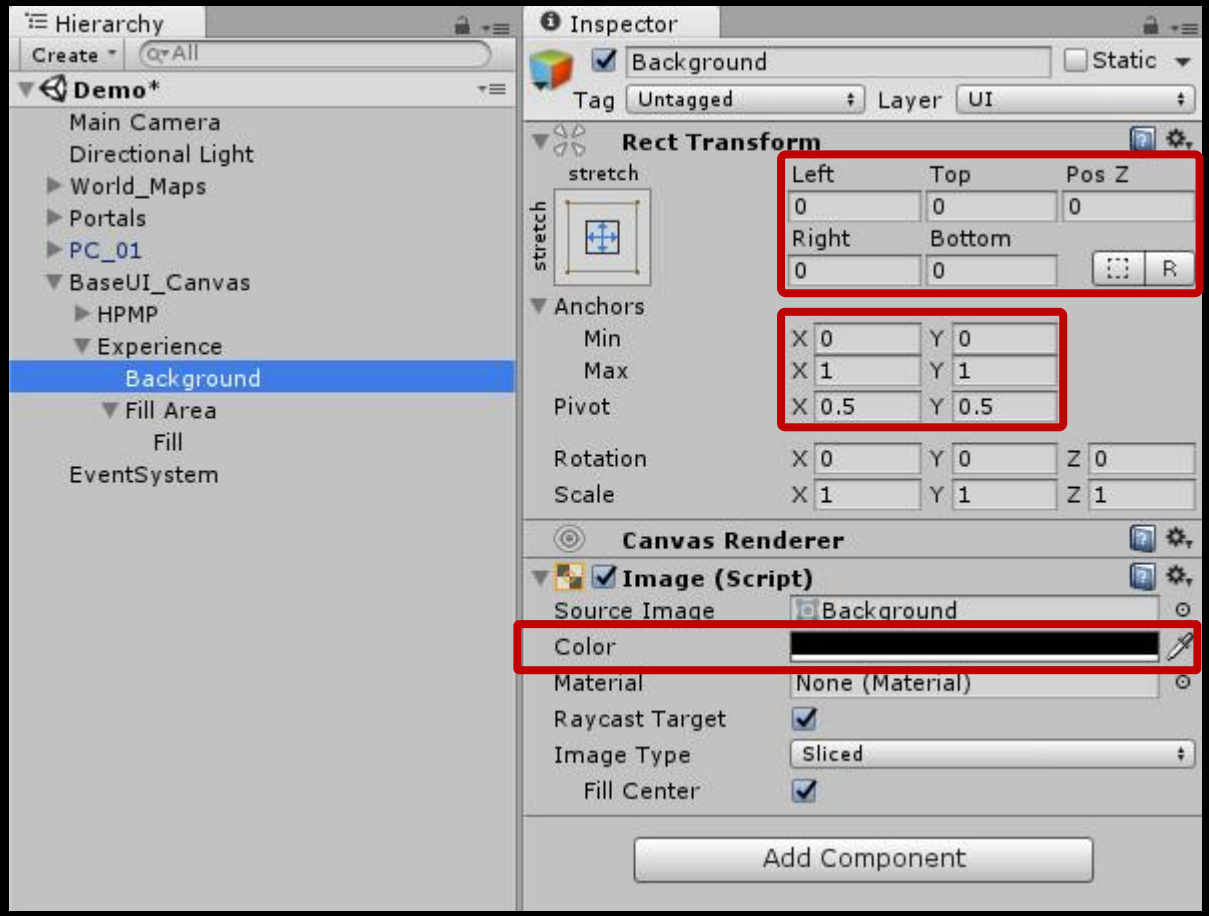
Base UI 설정

- Experience를 관리할 Slider 생성 및 설정(계속)



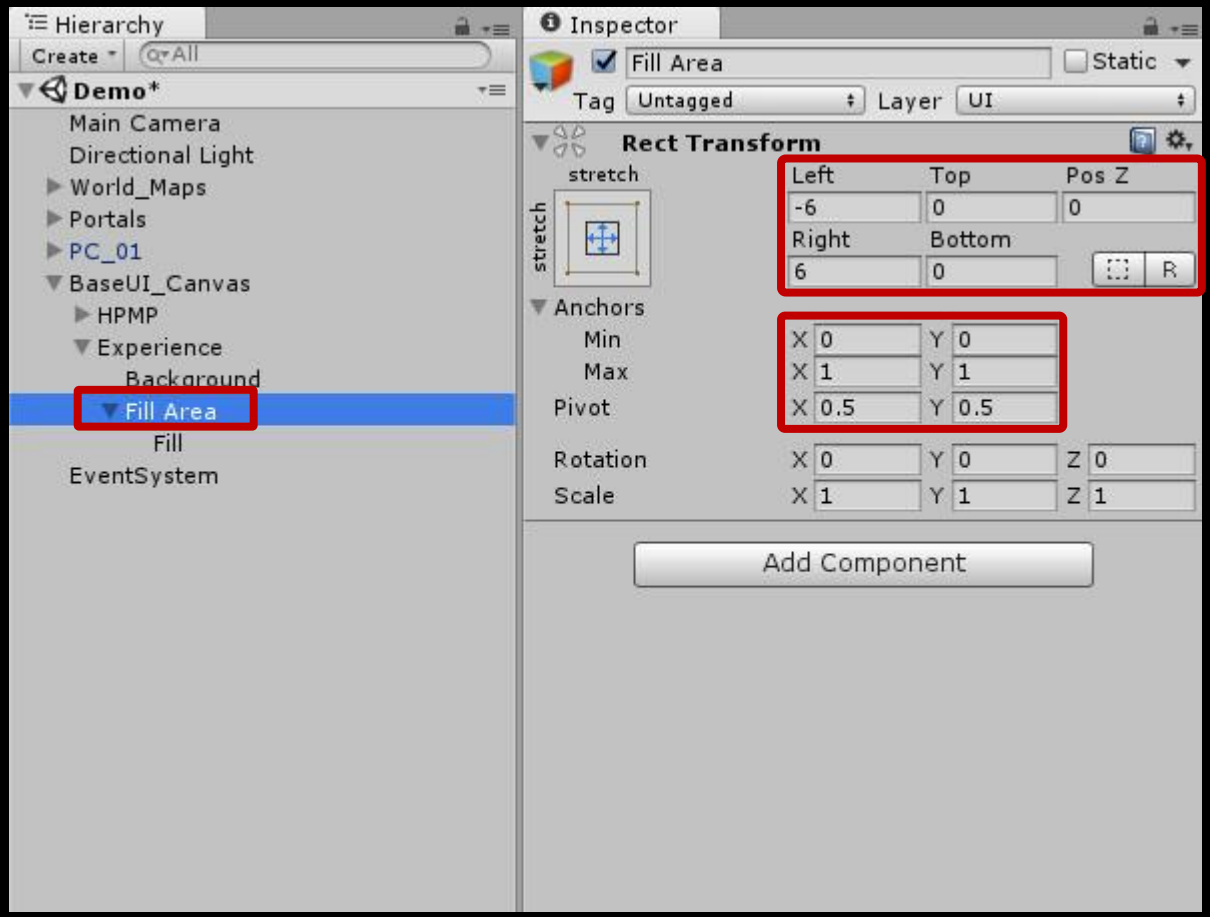
Base UI 설정

- Experience를 관리할 Slider 생성 및 설정(계속)
  - "Background"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 검은색으로 변경(알파값을 127로 설정)



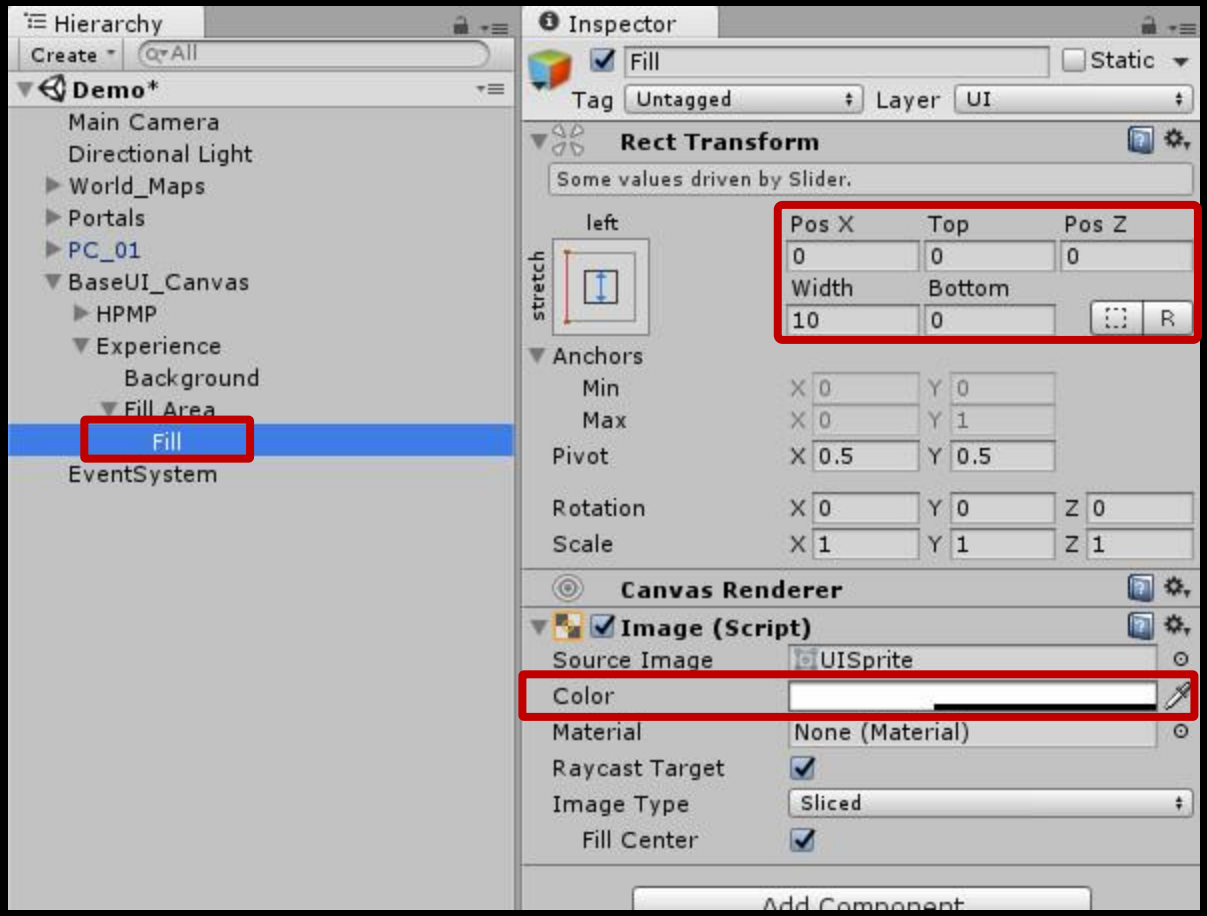
Base UI 설정

- Experience를 관리할 Slider 생성 및 설정(계속)
  - "Fill Area"의 Rect Transform 컴포넌트 설정



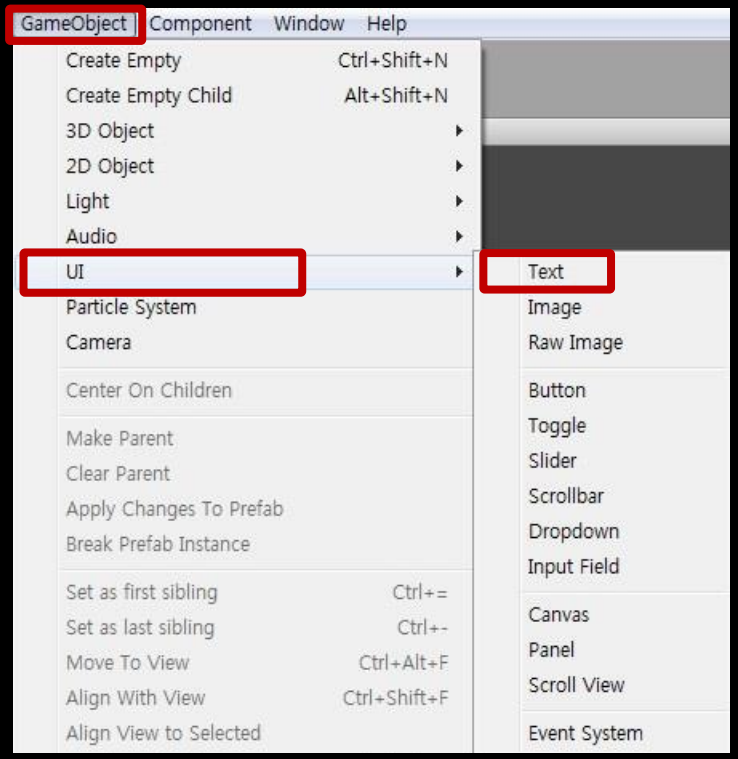
Base UI 설정

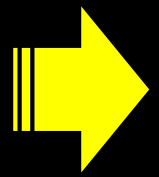
- Experience를 관리할 Slider 생성 및 설정(계속)
  - "Fill"의 Rect Transform 컴포넌트 설정
  - Image 컴포넌트에서 색을 하얀색으로 변경(알파값을 100으로 설정)

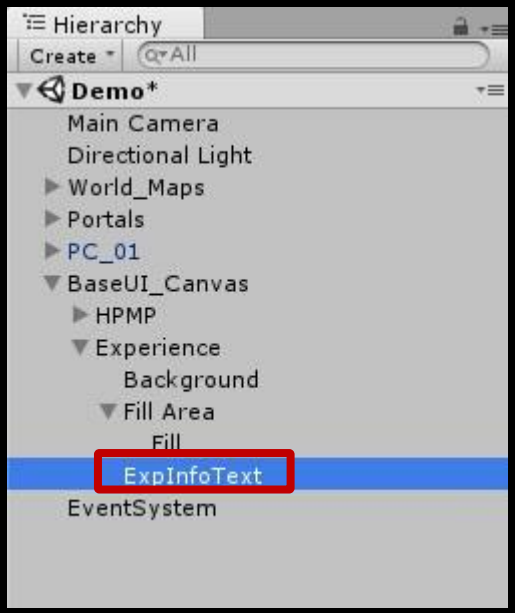


Base UI 설정

- 경험치 데이터를 표현할 Text 생성 및 설정
  - GameObject - UI - Text
  - 생성한 Text의 위치를 "Experience" Slider의 자식으로 변경
  - Text의 이름을 "ExpInfoText"로 변경

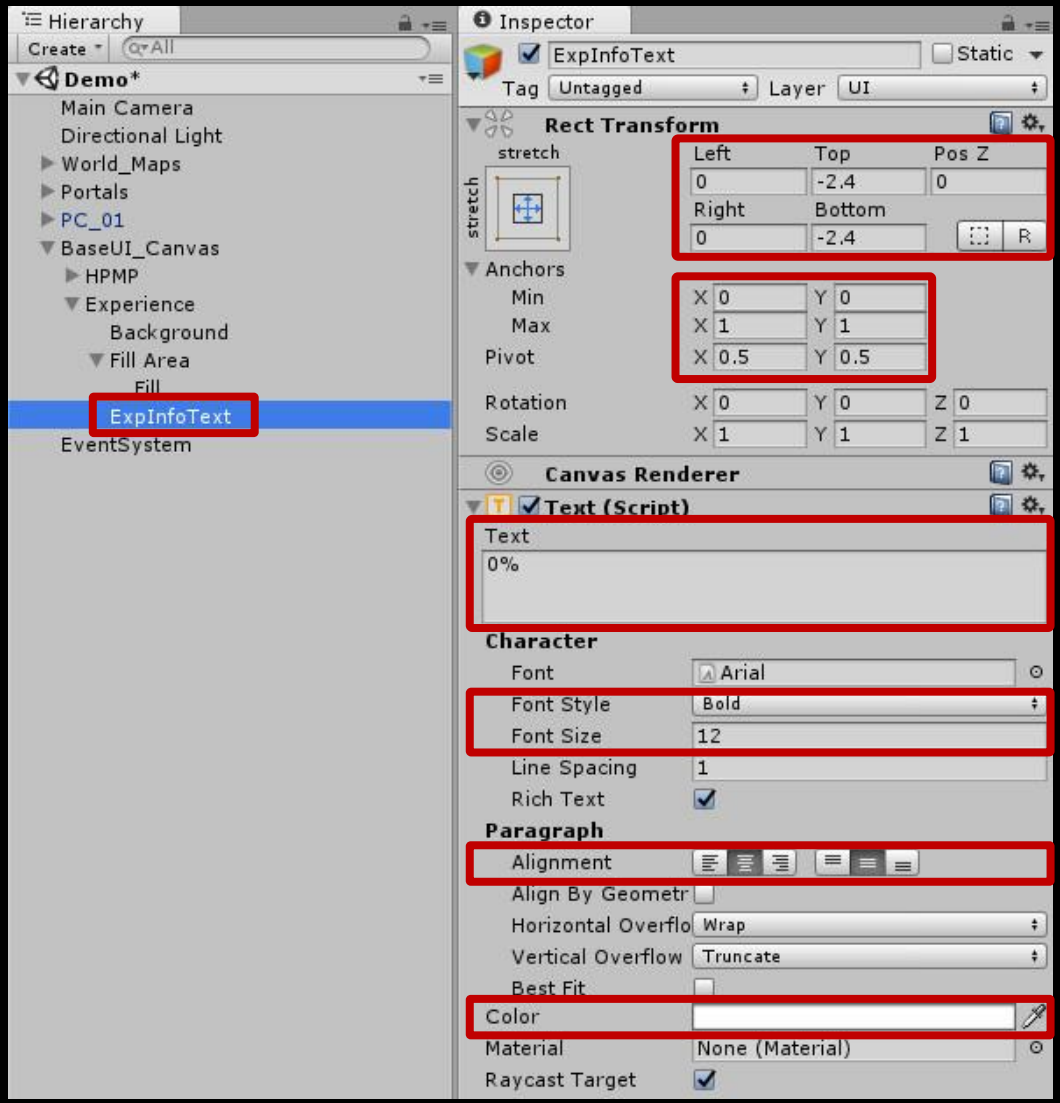






Base UI 설정

- 경험치 데이터를 표현할 Text 생성 및 설정(계속)



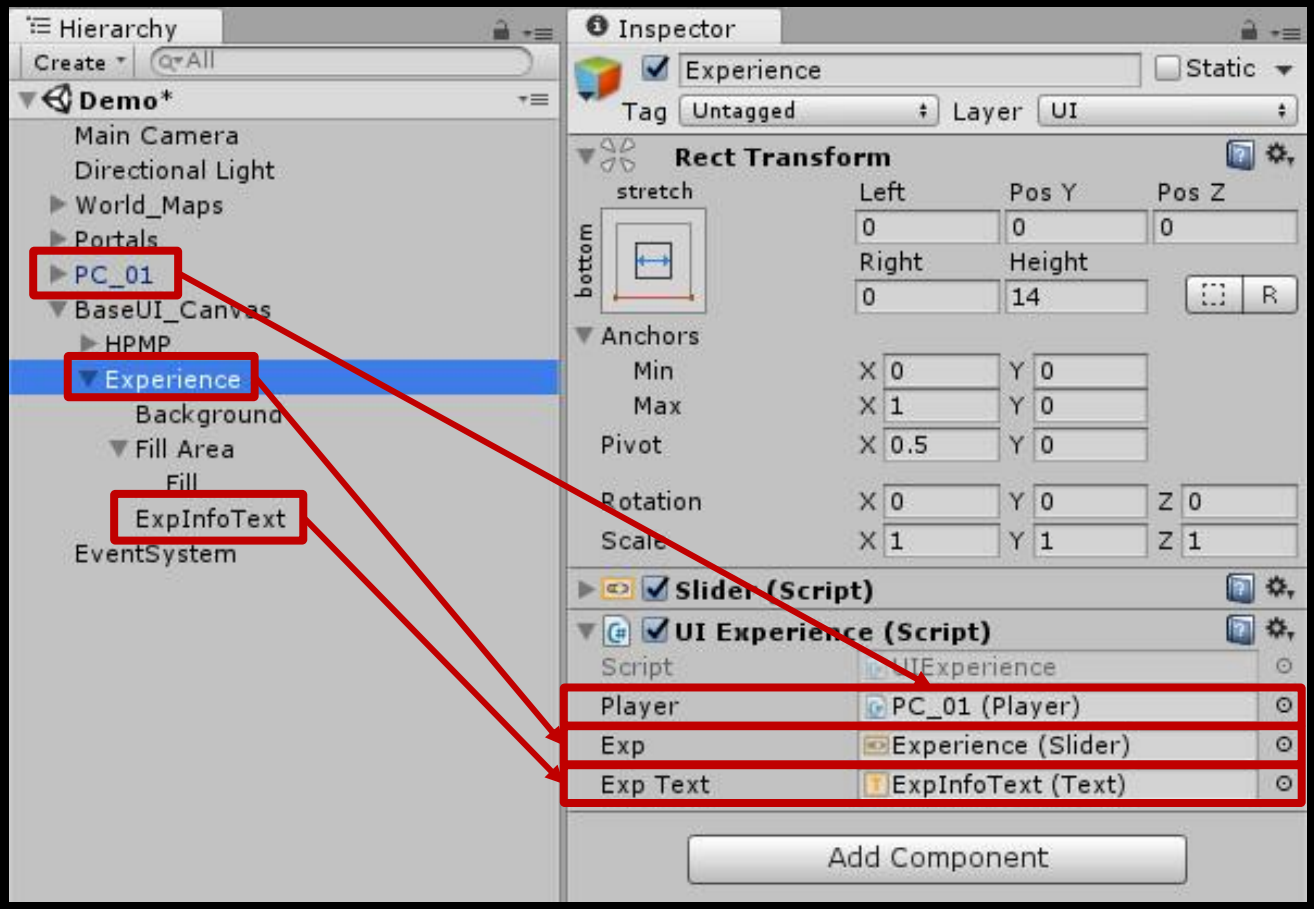
Base UI 설정

- 데이터와 UI 연결을 위한 스크립트 생성 및 작성
  - Project View - 마우스 오른쪽 클릭 - Create - C# Script
  - 생성한 스크립트의 이름을 "UIExperience"로 변경
  - 스크립트를 "Experience" Panel의 컴포넌트로 적용

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class UIExperience : MonoBehaviour
{
    [SerializeField] Player _player;
    [SerializeField] Slider _exp;
    [SerializeField] Text   _expText;

    void Update()
    {
        if ( _player.transform == null ) return;

        _exp.value      = _player.Exp_Percent();
        _expText.text   = "Lv." + _player.Level + " (" + (_player.Exp_Percent()*100.0f).ToString("F2") + "%)";
    }
}
```

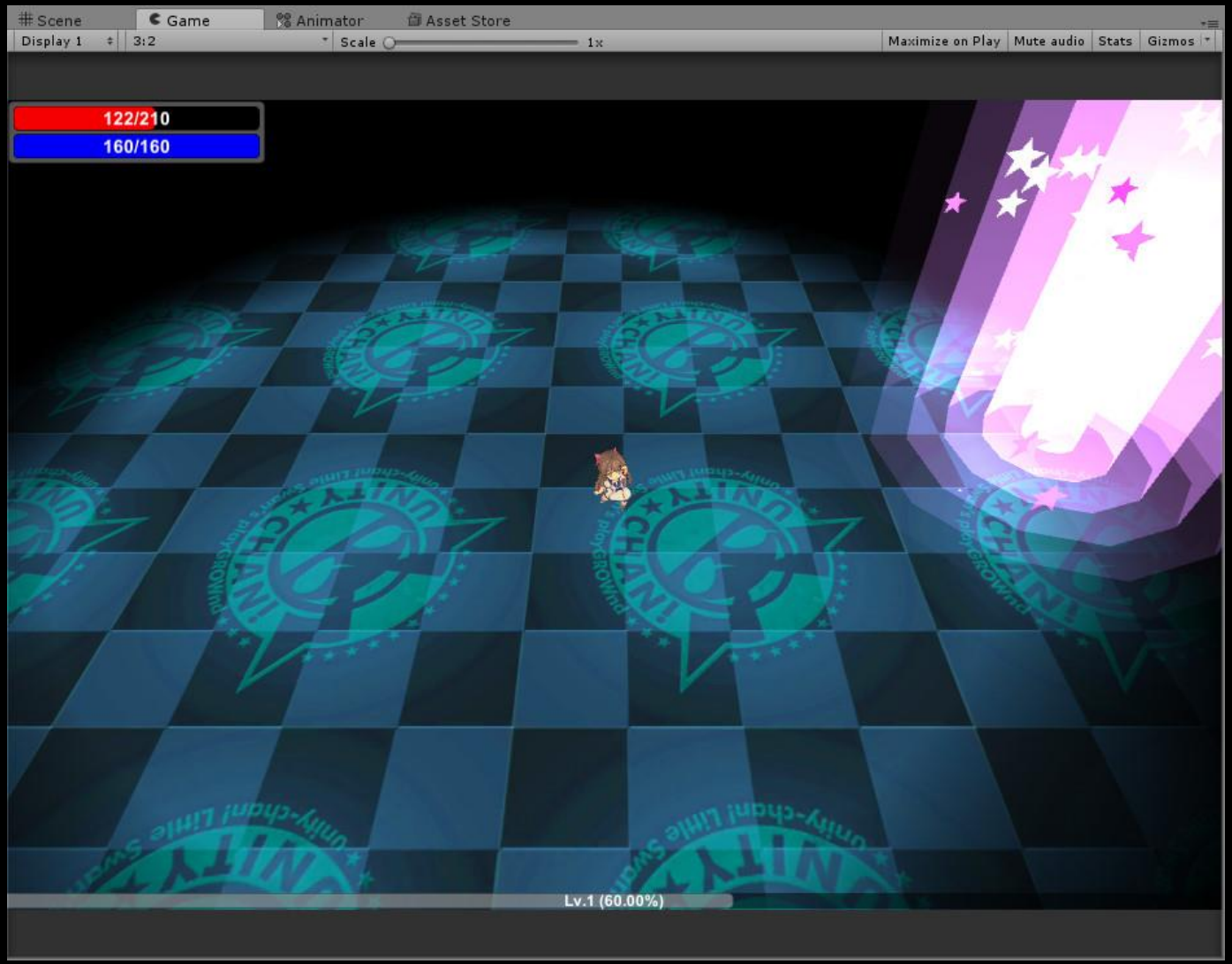
Base UI 설정

- UIExperience Script 변수 설정
  - Player 변수에 Exp 정보가 들어있는 캐릭터 클래스 저장
  - 정보를 변경할 Slider와 Text를 저장



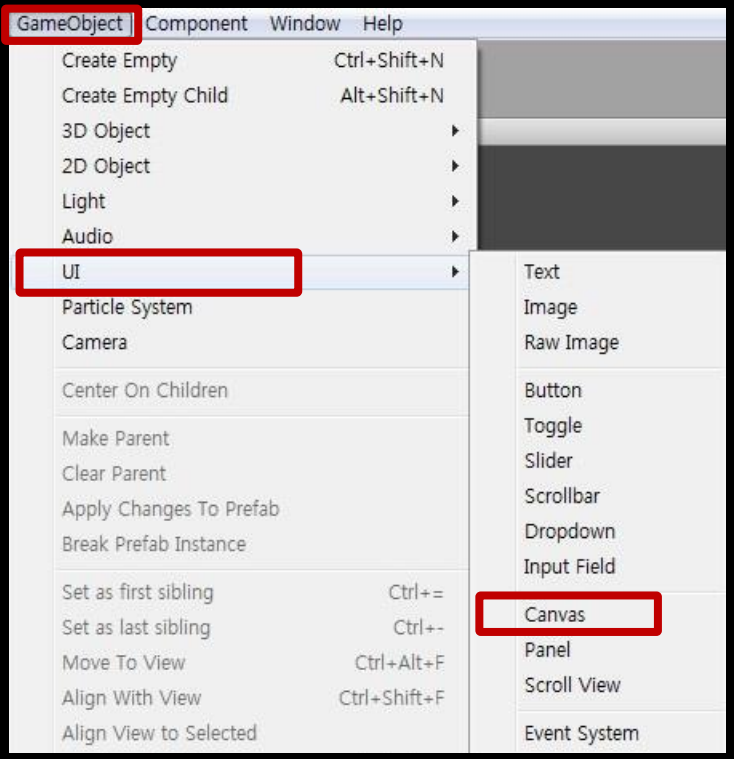
Base UI 설정

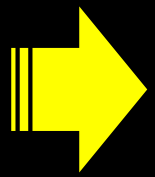
- 결과 화면

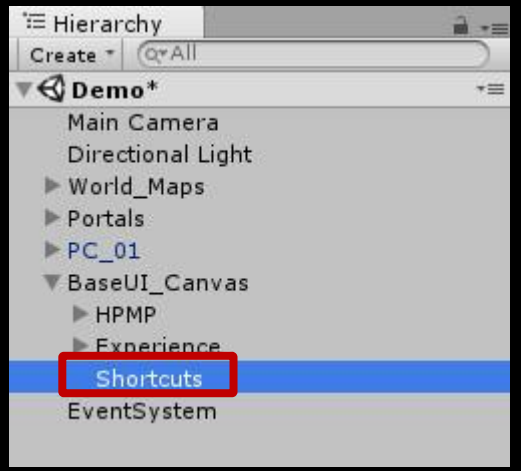


Base UI 설정

- **UIShortcuts**
  - Window UI들의 Icon을 가지고 있는 Panel 생성
    - GameObject - UI - Panel
    - 생성한 Panel의 이름을 "Shortcuts"로 변경

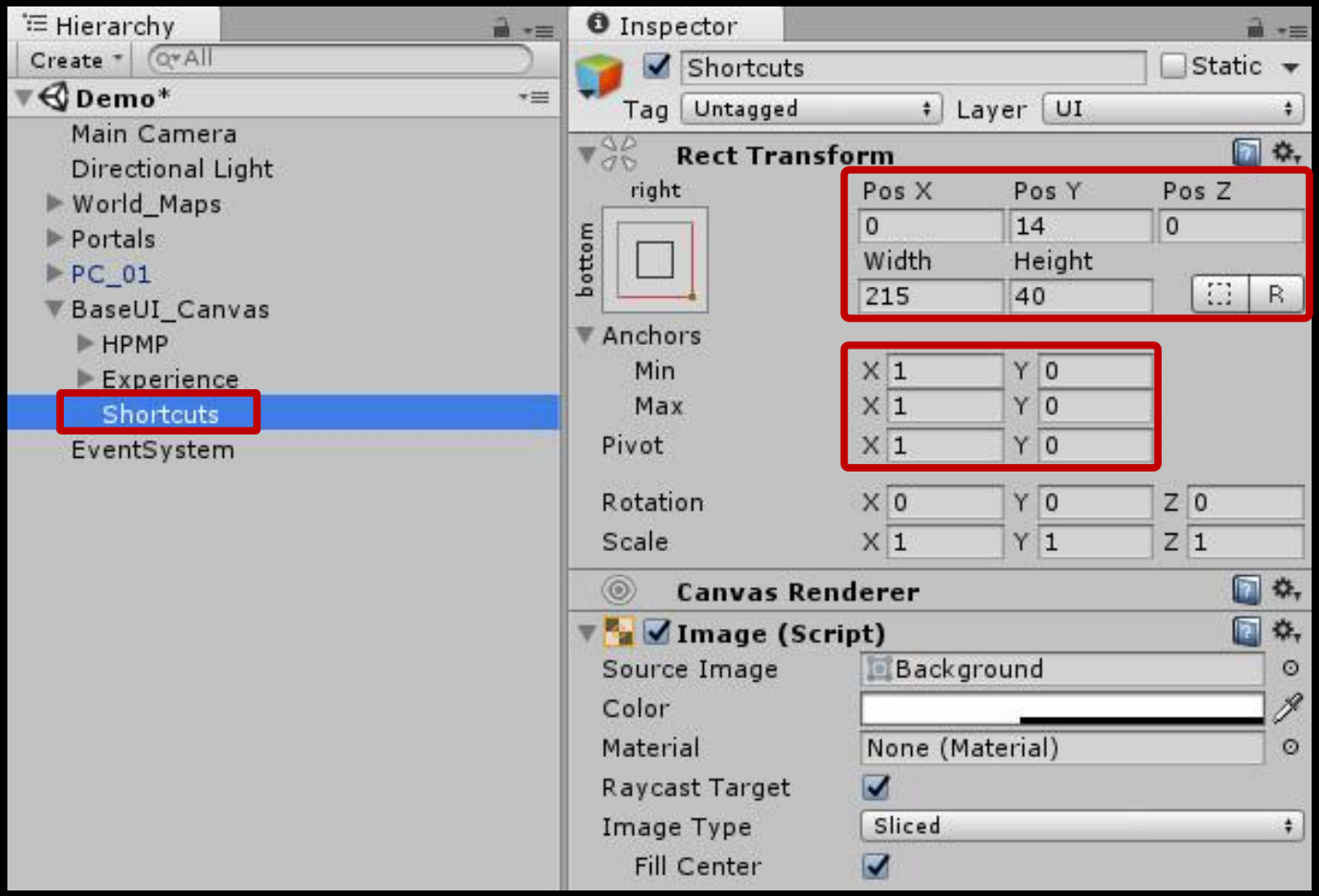






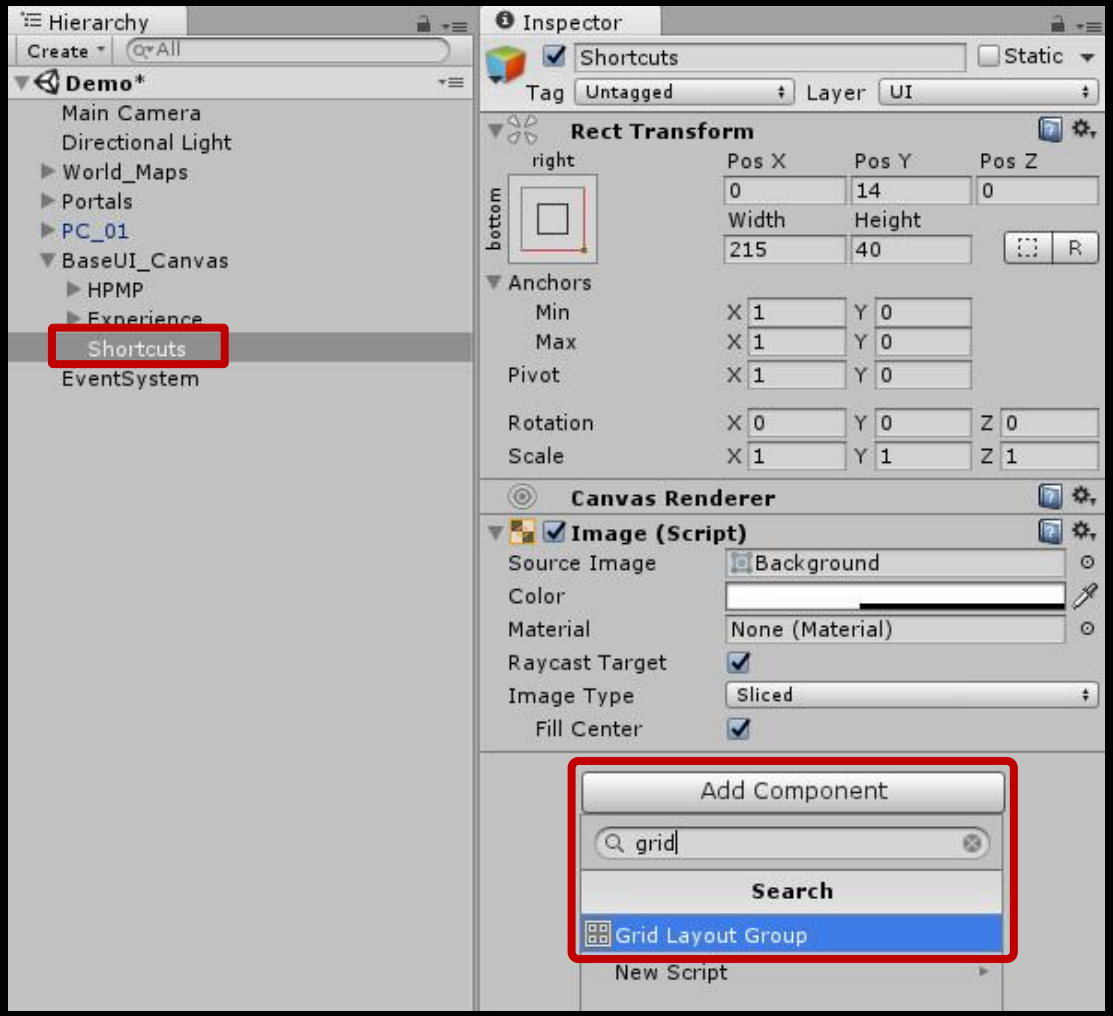
Base UI 설정

- Window UI들의 Icon을 가지고 있는 Panel 생성(계속)



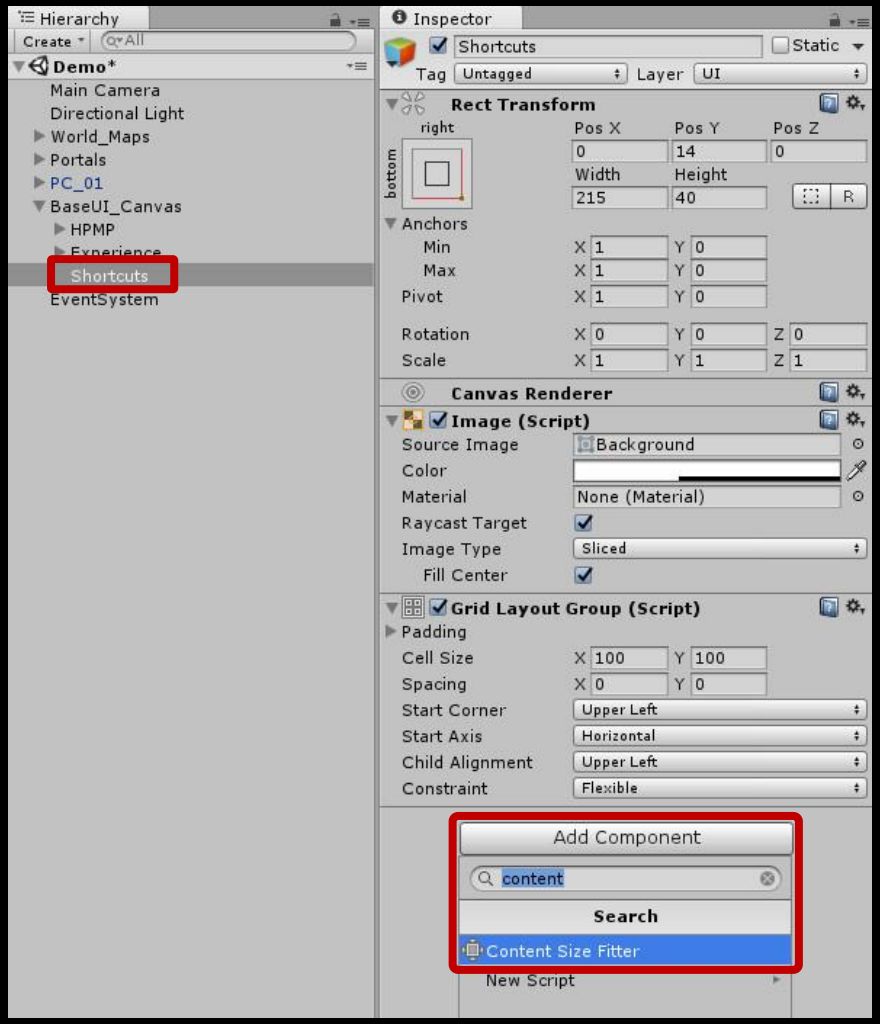
Base UI 설정

- Window UI들의 Icon을 가지고 있는 Panel 생성(계속)
  - Grid Layout Group 컴포넌트 추가



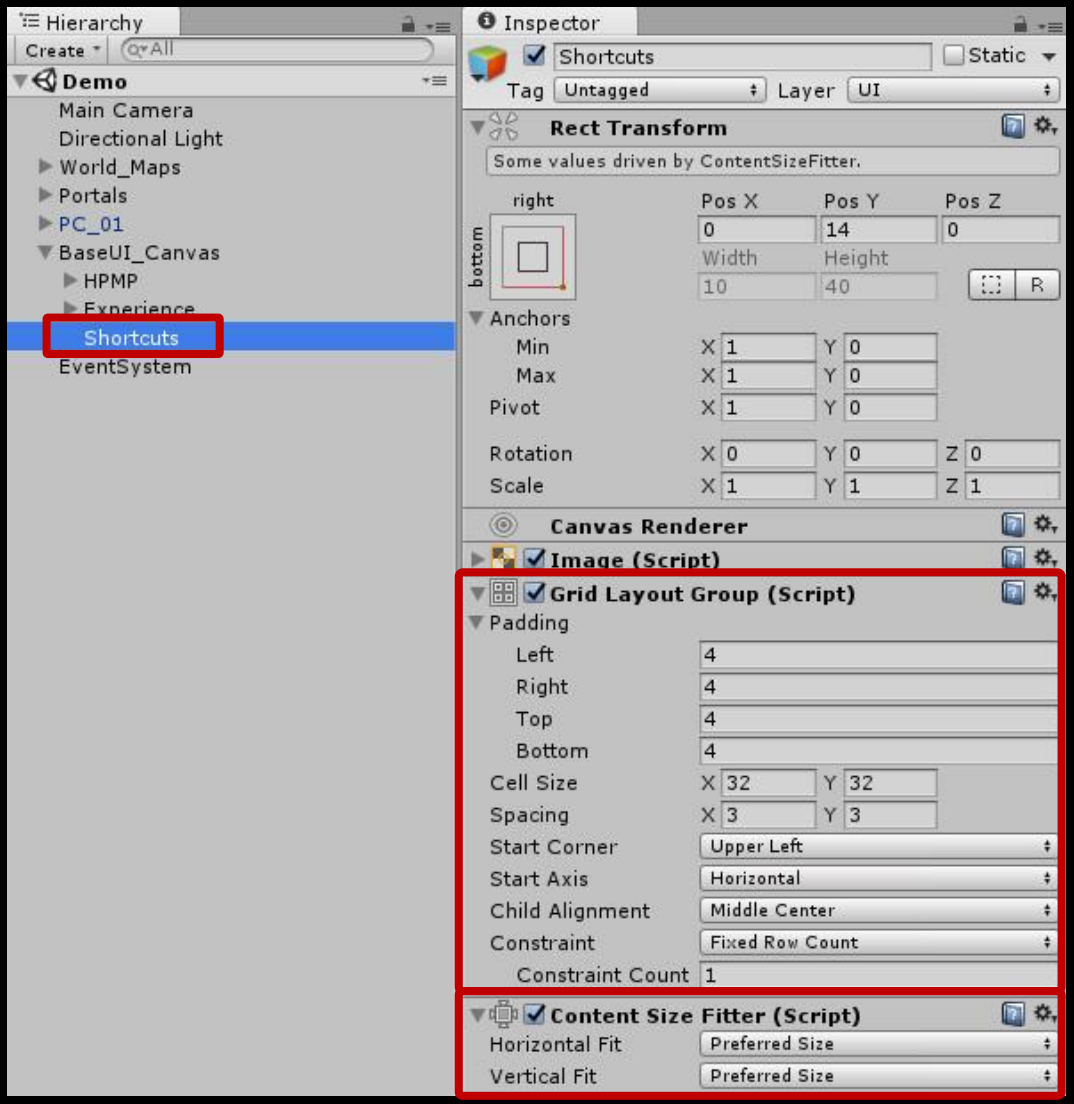
Base UI 설정

- Window UI들의 Icon을 가지고 있는 Panel 생성(계속)
  - Content Size Fitter 컴포넌트 추가



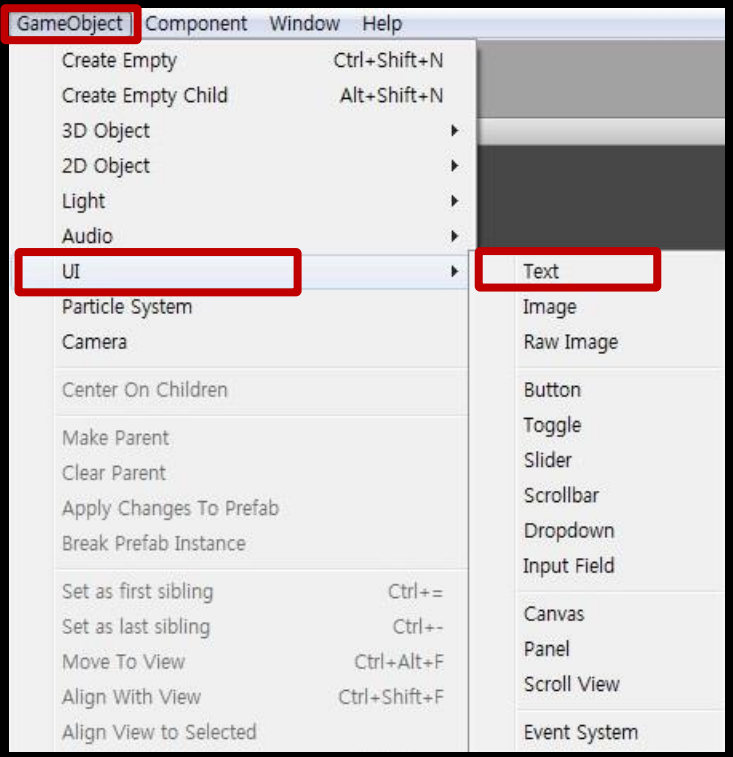
Base UI 설정

- Window UI들의 Icon을 가지고 있는 Panel 생성(계속)



Base UI 설정

- Icon으로 사용될 Button 생성
  - GameObject - UI - Button
  - 6개의 Button을 생성하고, 생성한 Button은 Shortcuts Panel의 하위에 저장
  - Button의 하위에 있는 Text는 사용하지 않으므로 삭제



Base UI 설정

- Icon으로 사용될 Button 생성(계속)
  - Button의 이름을 왼쪽에 위치한 것부터 순서대로 아래와 같이 변경한다.
    - Btn_Inventory, Btn_Equipment, Btn_Skills, Btn_Status, Btn_Quests, Btn_Exit
  - 각 Button의 Source Image에 해당 이름과 같은 그림을 적용한다.