UGUI에 SpriteAtlas를 사용하면서 불편한 사항중 하나인 아틀라스 별로 이미지를 보면서 Image컴포넌트에 세팅할 수 없다는 것. 그래서 툴을 만들기로 결정했다.

[목표]

1. 아틀라스 별로 이미지들을 볼 수 있어야 한다.

2. Image 컴포넌트에 넣을 수 있어야 한다.

3. 사용할 때 접근성이 좋고 편해야 한다.

툴 작업의 목표를 세웠고 기본적인 기능을 제외하고 제일 중요한 것은 역시 사용자의 편의성. 아무래도 가장 많이 사용할 UI 디자이너에게 친숙해야 툴로써 제 기능을 할 것이다. 그래서 NGUI의 GUI를 기본으로 했다. NGUI를 다뤄본 UI 디자이너가 많고 툴을 만드는데 회사에서 많은 시간을 할애할 수 없기 때문이다.

[작동방식]

1. Hierarchy에서 우클릭으로 스프라이트 세팅 창을 연다.

2. Hierarchy에서 Image를 가진 GameObject를 클릭했을 시 해당 GameObject가 Sprite를 세팅하는 대상이 된다.

3. SpriteAtlas를 세팅한다.

4. SpriteAtlas가 가진 이미지들을 스크롤에 표시한다.

5. 이미지를 더블 클릭했을 시 대상 GameObject의 Image에 선택한 이미지의 Sprite를 세팅한다.

일단 작동방식 부터 정하고. 필요한 부분을 하나씩 해결하자.

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public class SpriteSettingEditor : EditorWindow  {  static SpriteSettingEditor window;  static SpriteAtlas curAtlas; //선택한 아틀라스  static SpriteAtlas drawAtlas; //현재 보여주는 아틀라스  //아틀라스에서 얻어온 복사생성된 Sprite 배열  Sprite[] arrSprites;  //선택한 Sprite  Sprite selectedSprite;  //대상 GameObject의 Image  Image img\_current;  //Scroll 위치  Vector2 scroll\_pos;  //Double 클릭 체크용 시간 변수  float click\_delay = 0f;  [MenuItem("GameObject/Sprite Setting Tool", false, -100)]  public static void OpenWindow()  {  if(window == null)  window = CreateInstance<SpriteSettingEditor>();    window.Init(Selection.activeGameObject);  }  } |
| --- | --- |

일단 EditorWindow창을 띄우는 기본적인 세팅과 필요한 멤버 변수들을 선언하는데 SpriteAtlas들은 창을 껐다가 다 시켜도 같은 SpriteAtlas를 바라보기 위해 static으로 선언했다. 그리고 Hierarchy에서 우클릭했을 때 나오는 메뉴에 표시하기 위해 MenuItem("GameObject/") 하위로 어트리뷰트를 붙였다.

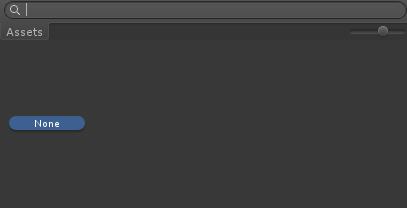
| 1  2  3  4  5  6  7  8  9  10 | /// <summary>  /// 아틀라스 및 이미지 선택 GUI를 그린다.  /// </summary>  void DrawAtlasAndImageSelector()  {  GUILayout.BeginHorizontal();  img\_current = (Image)EditorGUILayout.ObjectField(img\_current, typeof(Image), true, GUILayout.Width(150));  curAtlas = (SpriteAtlas)EditorGUILayout.ObjectField(curAtlas, typeof(SpriteAtlas), false);  GUILayout.EndHorizontal();  } |
| --- | --- |

이제는 선택된 오브젝트와 SpriteAtlas를 선택하는 GUI를 그릴 차래. 유니티 에디터 Layout의 좋은 기능인 ObjectField를 사용해서 해당 멤버 변수의 GUI를 그렸다.

*Object ObjectField(Object obj, Type objType, bool allowSceneObjects, params GUILayoutOption[] options);*

ObjectField로 레이아웃의 그리면 해당 타입의 파일을 찾기 기능이 있는데 allowSceneObjects가 true인 경우 Scene에서, false인 경우 Project에서 찾는다.

ObjectField로 GUI를 그렸을때 모습

ObjectField로 GUI를 그렸을때 파일 찾기 기능

| 1  2  3  4  5  6  7  8  9  10  11  12  13 | /// <summary>  /// 스프라이트 배열을 세팅한다.  /// </summary>  void SetSpritesArray(bool \_forced = false)  {  //지금 그리고 있는 아틀라스가 없거나 다른경우에 해당 아틀라스에서 스프라이트들을 얻는다.  if (\_forced == true || drawAtlas == null || drawAtlas != curAtlas)  {  drawAtlas = curAtlas;  arrSprites = new Sprite[curAtlas.spriteCount];  curAtlas.GetSprites(arrSprites);  }  } |
| --- | --- |

그다음은 세팅한 SpriteAtlas에서 Sprite 이미지들을 얻어온다. 이제 GUI에서 보여줘야할 기본적인 건 세팅이 끝났다. 얻어온 Sprite 이미지들을 GUI에서 스크롤로 보여주고 클릭 이벤트를 처리하면 작업 완료. 여기서 나는 GUI를 직접 꾸미기는 시간이 부족해서 NGUI의 SpriteSelector를 참고하여 GUI를 세팅했다.

NGUI의 SpriteSelector를 참고해서 만든 GUI

| 1  2  3  4  5  6  7  8  9  10  11  12 | if (GUI.Button(rect, ""))  {  selectedSprite = arrSprites[\_index];  float delta = Time.realtimeSinceStartup - click\_delay ;  click\_delay = Time.realtimeSinceStartup;  if (selectedSprite != null && delta < 0.5f)  {  SetSprite(selectedSprite.name.Replace("(Clone)", ""));  Close();  }  } |
| --- | --- |

버튼 같은 경우에는 GUI.Button으로 영역을 만들고 델타 타임을 얻어서 더블클릭 이벤트를 만들었고 동일한 영역에 이미지를 GUI.DrawTextureWithTexCoords을 이용하여 그렸다.

| 1  2  3  4  5  6  7  8  9  10 | /// <summary>  /// 스프라이트 이름으로 이미지를 세팅한다.  /// </summary>  /// <param name="\_spriteName"></param>  void SetSprite(string \_spriteName)  {  Sprite sprite = curAtlas.GetSprite(\_spriteName);  //SpriteAtlas의 GetSprite는 Clone으로 생성해서 반환하기 때문에 원본 텍스쳐를 접근해서 얻은 경로로 다시 Sprite를 로드한다.  img\_current.sprite = AssetDatabase.LoadAssetAtPath<Sprite>(AssetDatabase.GetAssetPath(sprite.texture));  } |
| --- | --- |

이제 더블 클릭을 했을때 Image 컴포넌트에 Sprite 이미지를 넣어주면 되는데 여기서 제일 중요한 문제가 처음에 발생했다. **SpriteAtlas에서 GetSprite로 이미지를 얻어오면 해당 Sprite 이미지를 복사 생성해서 들고 온다는 것.** 그래서 세팅한 프리펩을 불러왔을 때 Image 컴포넌트가 비어있는 문제가 발생하게 된다.

내가 문제를 해결한 방법은 어차피 툴이라서 인게임에서 돌 것도 아니라서 복사 생성된 Sprite 이미지의 texture를 통해서 해당 이미지의 파일 경로를 얻어와서 AssetDatabase.LoadAssetAtPath를 통해 원본 Sprite 이미지를 Image 컴포넌트에 꼽아줬다.