Dear IMGUI를 유니티로 통합하기

2019-03-08

Kiyoung Moon, ProgC

Giantstep

# Dear imgui란?

Dear imgui는 immediate gui 방식으로 구성된 GUI 라이브러리다. retained gui와 동작하는 방식이 다르며 더 자세한 설명은 좋은 영상이 있으므로 영상 링크로 대체한다.

<https://www.youtube.com/watch?v=LSRJ1jZq90k&feature=youtu.be>

# 유니티에 통합하기

Dear imgui는 C++로 구성된 크로스플랫폼 gui 라이브러리이며 안타깝게도 유니티에 통합된 사례는 없다. 유니티에 통합하기 위해서 C++로 제작된 유니티 플러그인이 필요하며 유니티에서 플러그인 함수를 호출하기 위해서 C#으로 작성된 구동 코드가 필요하다.

# C++로 유니티플러그인 작성

먼저 C++로 작성된 유니티 플러그인 작성이 필요하다. 아래 주소에 들어가서 Low-level native plug-in interface에 관련된 문서를 읽도록 한다.

<https://docs.unity3d.com/Manual/NativePluginInterface.html>

문서의 마지막 부분을 보면 <https://bitbucket.org/Unity-Technologies/graphicsdemos> 를 볼 수 있는데 이 부분을 참고하자.

공식 문서로도 충분하지만 설명을 덧붙이자면 기본적으로 유니티가 dll을 로드하게 되면 UnityPluginLoad라는 함수를 호출하며 플러그인이 언로드 되면 UnityPluginUnload라는 함수가 호출된다. 그래픽스 디바이스에 대한 정보를 얻기 위해서 콜백 함수를 등록해주어야 한다.

|  |
| --- |
| void UnityPluginLoad(IUnityInterfaces\* UnityInterfacesPtr)  {  UnityInterfaces = UnityInterfacesPtr;  UnityGraphics = UnityInterfaces->Get<IUnityGraphics>();    UnityGraphics->RegisterDeviceEventCallback(OnGraphicsDeviceEvent);  OnGraphicsDeviceEvent(kUnityGfxDeviceEventInitialize);  } |

이 문서에서 공개한 플러그인은 DirectX11만을 지원하며 DirectX12 혹은 OpenGL을 지원하고 싶다면 파이팅을 외쳐주고 싶다. ☺

# 윈도우 핸들 얻기

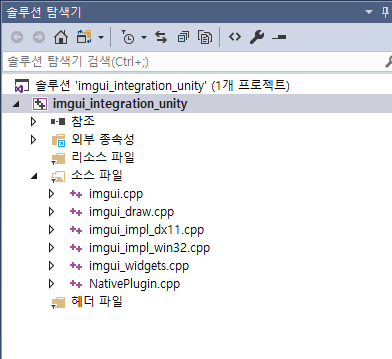
유니티에서 imgui가 제대로 동작하려면 윈도우 핸들을 넘겨야 하는데 에디터 모드일 때와 빌드 된 프로그램에서 핸들을 얻는 방법이 다르다. 이것을 해결하기 위해서는 유니티에서 플러그인을 로드한 후 에디터 모드인지 전체화면 모드인지 알려주거나 하는 다양한 방법이 있을 수 있지만 내가 선택한 것은 단순히 전처리기를 설정하고 두 개의 별도의 dll을 사용하는 것으로 작성했다.

|  |
| --- |
| // get unity window handle.  #ifdef UNITY\_EDITOR  *HWND* rootHandle = *FindWindow*(L"UnityContainerWndClass", nullptr);  if (rootHandle != nullptr)  {  unityHandle = *FindWindowEx*(rootHandle, nullptr,  L"UnityGUIViewWndClass", L"UnityEditor.GameView");  }  else  {  *MessageBox*(rootHandle, *TEXT*("Can't found rootHandle"),  *TEXT*("Caption"), *MB\_OK*);  }  #else  unityHandle = *GetActiveWindow*();  #endif |

전체 화면일 때는 GetActiveWindow를 사용하고 에디터 모드일 때는 클래스 이름과 게임 뷰 이름을 검색해서 윈도우 핸들을 찾는다. 이 부분에 있어서는 좀 더 리서치가 필요하지만 해야할 다른 일들이 많으므로 더 좋은 방법은 해결하신분이 공유해 주었으면 좋겠다.

# imgui를 위한 프로젝트 설정

Imgui를 유니티 플러그인에서 사용하기 위해서 다음과 같이 소스를 구성해야 한다.



# 윈도우 메시지 처리

imgui에서 키보드 및 마우스와 같은 메시지를 처리하기 위해서 윈도우 메시지를 전달할 필요가 있기 때문에 메시지 프로시저를 가로채야 한다. 이를 위해서 다음과 같은 처리가 필요하다.

|  |
| --- |
| // get wndproc  oldWndProc = (*WNDPROC*)*SetWindowLongPtr*(unityHandle, *GWLP\_WNDPROC*, (*LONG\_PTR*)MyWndProc); |

unityHandle로부터 WndProc을 내가 정의한 MyWndProc으로 재정의한다. 그리고 MyWndProc에서 메시지를 받으면 imgui로 전달하고 나머지 메시지들은 본래 유니티 메시지 처리기에 넘긴다.

|  |
| --- |
| *LRESULT* *CALLBACK* MyWndProc(*HWND* hWnd, *UINT* message, *WPARAM* wParam, *LPARAM* lParam)  {  if (ImGui\_ImplWin32\_WndProcHandler(hWnd, message, wParam, lParam))  return true;    // otherwise let unity process it.  return *CallWindowProc*(oldWndProc, hWnd, message, wParam, lParam);  } |

# Imgui 그리기

|  |
| --- |
| static void UNITY\_INTERFACE\_API OnRenderEvent(int eventID)  {  ImGui\_ImplDX11\_NewFrame();  ImGui\_ImplWin32\_NewFrame();  ImGui::NewFrame();  ImGui::Begin("GXLab");  ImGui::Text("Face weights");  ImGui::PlotHistogram("weights", facialWeights, facialWeightCount, 0, *NULL*, 0.0f, 100.0f, ImVec2(0, 150));  ImGui::Text("Expression : Happy");  ImGui::Text("MousePos : %d %d", (int)ImGui::GetIO().MousePos.x, (int)ImGui::GetIO().MousePos.y);  ImGui::End();    ImGui::Render();    ImGui\_ImplDX11\_RenderDrawData(ImGui::GetDrawData());  }  extern "C" UnityRenderingEvent UNITY\_INTERFACE\_EXPORT UNITY\_INTERFACE\_API GetRenderEventFunc()  {  return OnRenderEvent;  } |

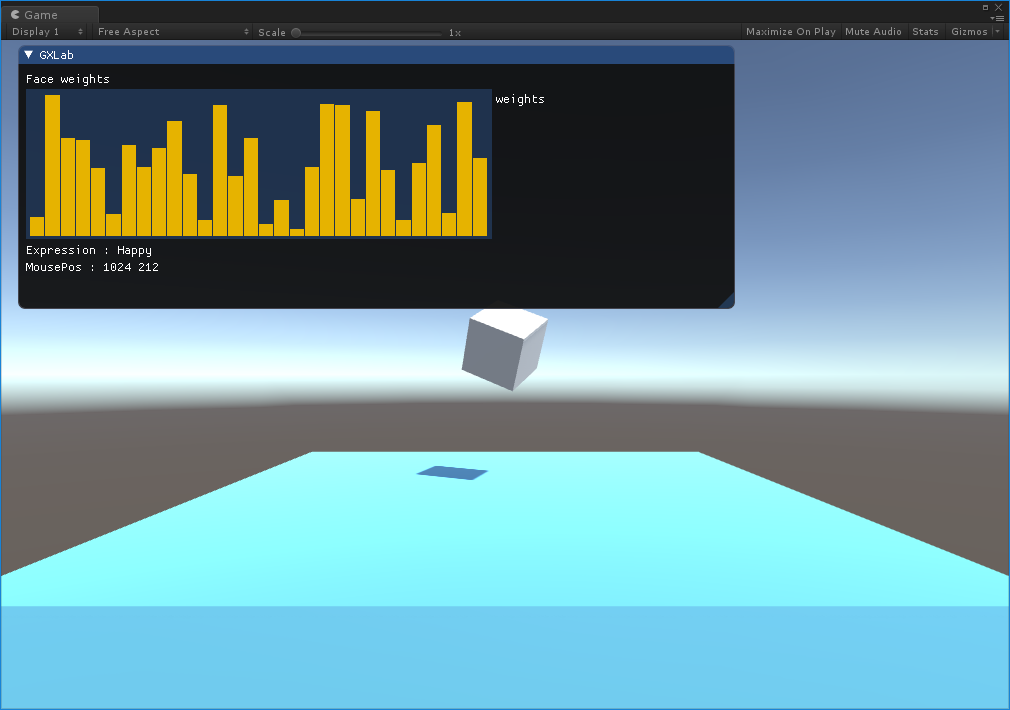
유니티의 C#코드에서 GetRenderEventFunc을 렌더링 커맨드로 넣어주고 이 안에서 OnRenderEvent를 호출하며, 그 안에서 imgui을 사용해 화면에 그림을 그려준다.

# 유니티에서 플러그인 호출하기

만들어진 dll들을 유니티에서 호출하기 위해서는 구동 코드가 필요하다.

|  |
| --- |
| using *UnityEngine*;  using *System*;  using *System*.*Collections*;  using *System*.*Runtime*.*InteropServices*;  public class imgui\_integration\_unity : *MonoBehaviour* {  #if UNITY\_EDITOR  [DllImport("imgui\_integration\_unity\_Editor")]  #else  [DllImport("imgui\_integration\_unity")]  #endif  static extern *IntPtr* GetRenderEventFunc();  #if UNITY\_EDITOR  [DllImport("imgui\_integration\_unity\_Editor")]  #else  [DllImport("imgui\_integration\_unity")]  #endif  public static extern void SetFacialWeights(int count, *IntPtr* weights);  *IEnumerator* Start()  {  yield return *StartCoroutine*("CallNativePlugin");  }  *IEnumerator* CallNativePlugin()  {  while (true)  {  yield return new *WaitForEndOfFrame*();  *GL*.*IssuePluginEvent*(GetRenderEventFunc(), 1);  }  }  } |

특이점이라면 UNITY\_EDITOR일 때는 \*\_Editor.dll을 로드하며 아닐 때는 전체 화면용 dll을 로드하도록 했다. 이렇게 만들어진 스크립트를 아무 게임 오브젝트에 부착하고 실행하면 GetRenderEventFunc가 호출되며 내부적으로 imgui 위젯이 화면에 그려지게 된다.



# 마지막으로

유니티와 플러그인 소스는 github에 올려두었습니다.

개발자 뽑습니다. <http://www.giantstep.co.kr/view/recruit.php> 연락주세요~