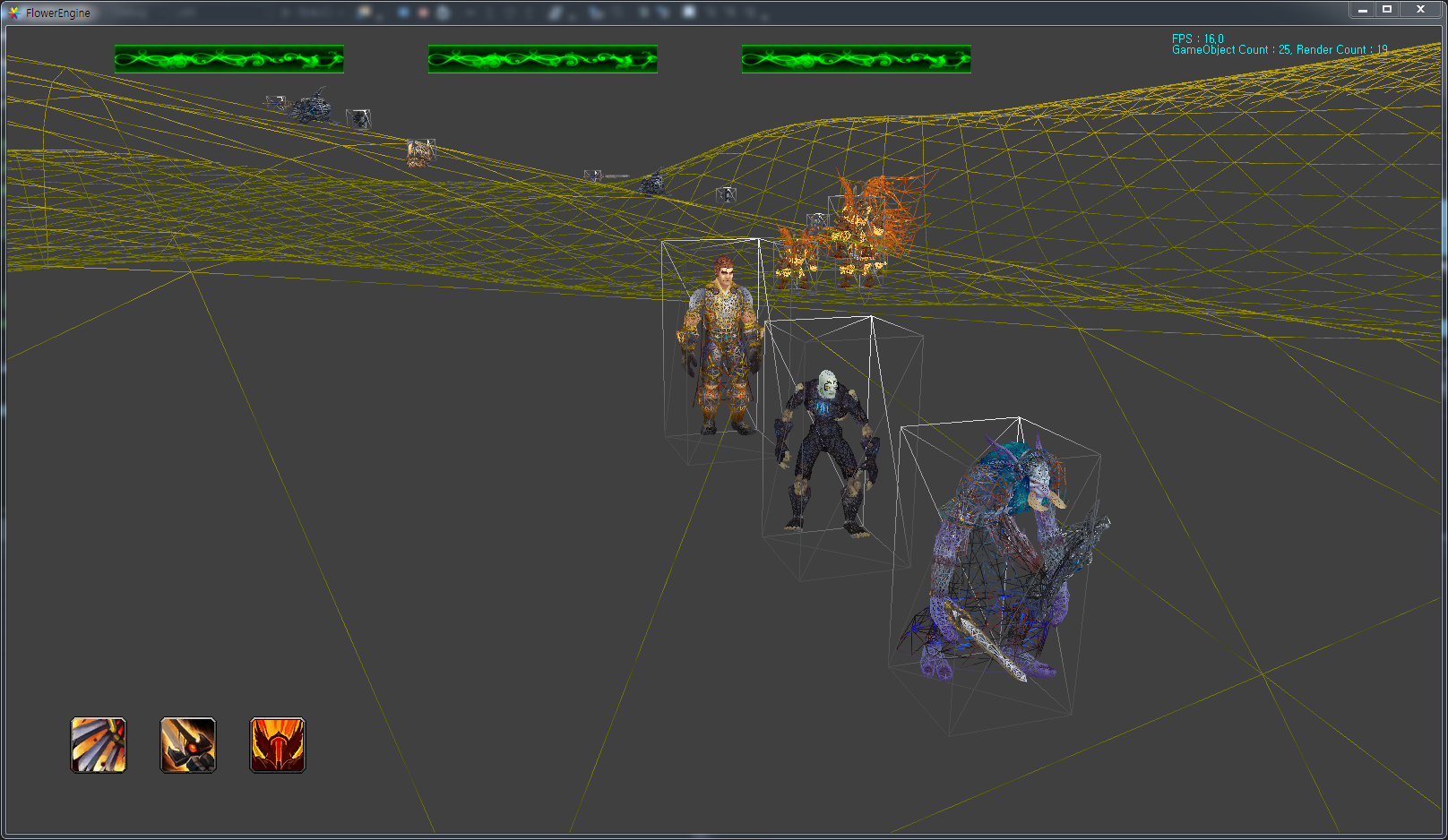
**포트폴리오**

**클라이언트 프로그래머**



**작성자 : 송현국**

[hkn10004@naver.com](mailto:hkn10004@naver.com)

HP : 010-3592-5921

팀원 : 김동오, 이호형

선생님 : 서울게임아카데미(SGA) 김주안

목차

1장 : 전체적인 설명

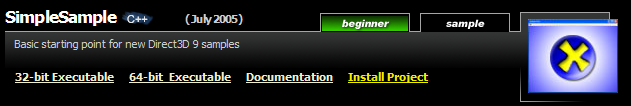
2장 : 전투 개발

3장 : 전투 외 개발

게임 개발의 시작

저는 경력 5년이상 된 프로그래머입니다. 개인적으로 공부를 더 하고 안정적인 개발자가 되기 위해서 국비지원 수업을 찾던 중 서울게임아카데미의 수업과정을 보고 신청하게 되었고 DirectX API를 이용하여 개발하게 되었습니다. 엔진은 언리얼, 유니티, Cocos2D-X 어떠한 엔진을 써도 상관은 없으나 함께 공부하는데에 DirectX를 사용하므로 자체엔진을 개발하게 되었습니다.

기반 윈도우 프레임워크



일단, DXUT를 사용하면 DirectX Sample에서 제공하는 UI등 많은 것들을 사용할 수 있으므로 DirectX Samples에서 제공하는 SimpleSample을 사용하였습니다. 물론, 이 예제들에서 제공하는 것들을 가공해야 합니다.

프로그램의 시작 WinMain

콜백함수란 함수 포인터로 프레임워크에 필요한 함수를 재정의 해서 그 함수를 연결하는 것입니다. DXUT에서 필요한 부분에 연결한 함수가 호출됩니다.

콜백함수를 간단히 설명하자면 윈도우 메시지 처리, 키보드 마우스 입력 처리, 프레임 업데이트, 렌더링, 디바이스 세팅 관련 콜백함수들이 있습니다.

|  |
| --- |
| //--------------------------------------------------------------------------------------  // Entry point to the program. Initializes everything and goes into a message processing  // loop. Idle time is used to render the scene.  //--------------------------------------------------------------------------------------  int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR lpCmdLine, int nCmdShow)  {  // Set DXUT callbacks  DXUTSetCallbackMsgProc(MsgProc);  DXUTSetCallbackKeyboard(OnKeyboard);  DXUTSetCallbackFrameMove(OnFrameMove);  DXUTSetCallbackDeviceChanging(ModifyDeviceSettings);  DXUTSetCallbackD3D9DeviceAcceptable(IsD3D9DeviceAcceptable);  DXUTSetCallbackD3D9DeviceCreated(OnD3D9CreateDevice);  DXUTSetCallbackD3D9DeviceReset(OnD3D9ResetDevice);  DXUTSetCallbackD3D9DeviceLost(OnD3D9LostDevice);  DXUTSetCallbackD3D9DeviceDestroyed(OnD3D9DestroyDevice);  DXUTSetCallbackD3D9FrameRender(OnD3D9FrameRender);  InitApp();  DXUTInit(true, true, NULL);  DXUTSetCursorSettings(true, true);  DXUTCreateWindow(L"FlowerEngine");  DXUTCreateDevice(true, 1600, 900);    Camera::GetInstance()->Init();  SceneManager::GetInstance()->AddScene(new SceneRPG("SceneRPG"));  /\*SceneManager::GetInstance()->AddScene(new SceneAircraft("SceneAircraft"));  SceneManager::GetInstance()->AddScene(new ScenePicking("ScenePicking"));\*/  SceneManager::GetInstance()->ChangeScene("SceneRPG");  SceneManager::GetInstance()->Awake();    DXUTMainLoop(); // Enter into the DXUT render loop  return DXUTGetExitCode();  } |

GameObject Component System 설계

많은 분께서 아시다시피 GameProgramming Gems 6권에도 소개되어 있고 유니티, 언리얼등에서 사용하는 설계방식 입니다. 기본 내용을 공부하여 코드 작성 하였습니다.

간략히 설명 드리겠습니다. 위 코드에 보면 SceneManager를 사용하여 시작하는 것을 볼 수 있습니다. SceneManager는 Scene 관리를 담당하는 싱글턴 클래스입니다. 씬을 추가하거나 뺄 수 있고 변경할 수 있습니다. 그리고 현재 씬 객체를 가져오는 함수가 있습니다. 씬에는 추가한 GameObject 객체들이 있고 GameObject에는 구성요소(Component)들이 포함되어 있습니다.

그림(UML)으로 설명하자면 다음과 같습니다.

