[ 게임 UI/UX 프로그래밍 ]

**UI 프레임워크 및 아키텍처 구현**

인하대학교 미래인재개발원 문화컨텐츠개발

김예슬

**UI 프레임워크 개발 설계 문서**

- 맵툴

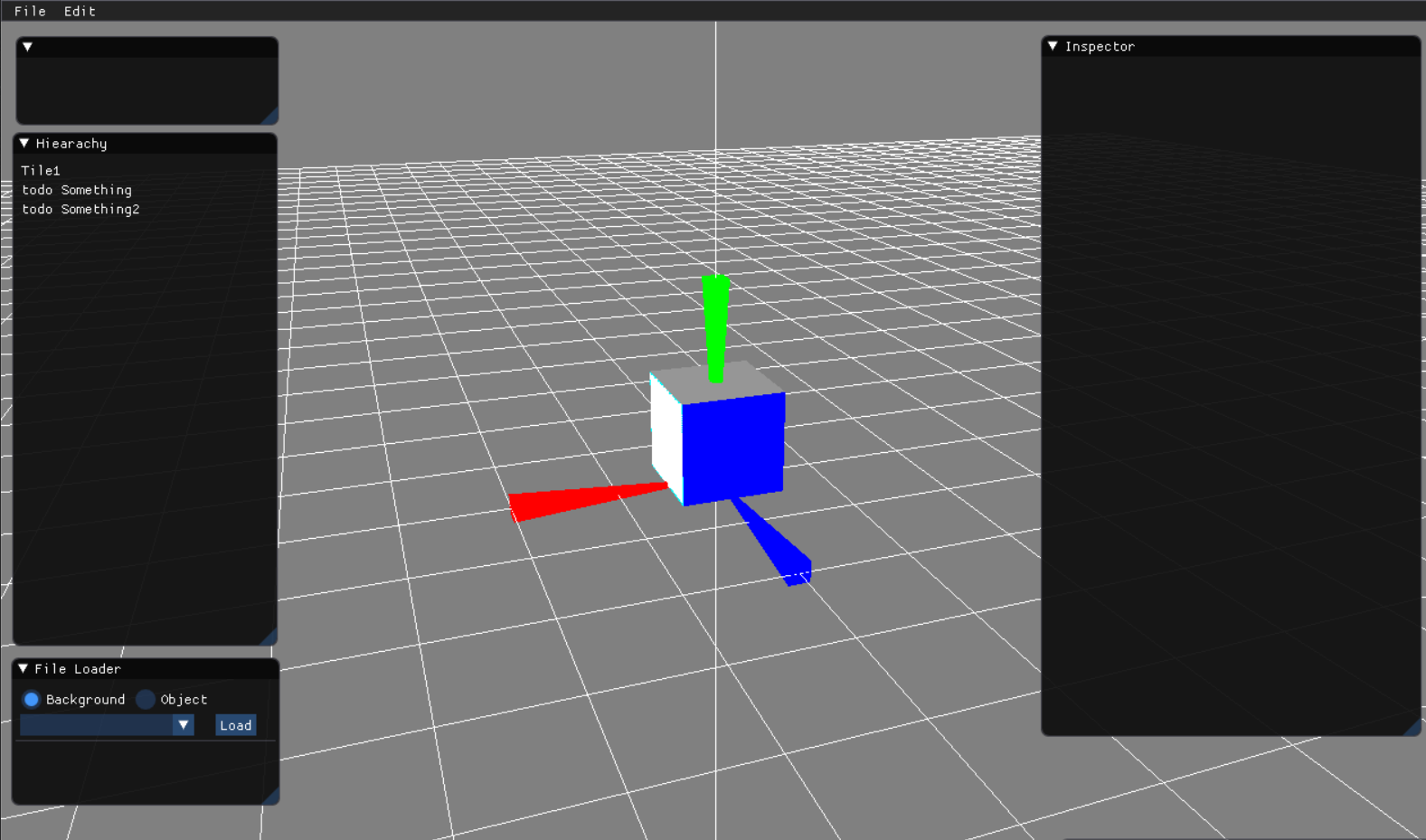
IMGUI 이용

파일을 로드 할 수 있는 작업창이 있어야 함

생성한 파일 리스트가 존재함(Hierarchy)

리스트에서 현재 누른 오브젝트의 정보 값을 확인 할 수 있어야 함(Inspector)

**완성된 UI 스크린 샷**

****

**완성된 UI 프로그래밍 코드**

{ // >> : Menu Title Bar

static bool isReset = false;

if (ImGui::BeginMainMenuBar())

{

if (ImGui::BeginMenu("File"))

{

if (ImGui::MenuItem("Open", " ")) { g\_pFileLoadManager->FileLoad\_OpenMapData(); }

if (ImGui::MenuItem("Save", " ")) { g\_pFileLoadManager->FileLoad\_SaveMapData(); }

ImGui::Separator();

if (ImGui::MenuItem("Reset", " "))

isReset = true;

ImGui::EndMenu();

}

if (ImGui::BeginMenu("Edit"))

{

// if (ImGui::MenuItem("Undo", "CTRL+Z")) {}

// if (ImGui::MenuItem("Redo", "CTRL+Y", false, false)) {}

// ImGui::Separator();

// if (ImGui::MenuItem("Cut", "CTRL+X")) {}

// if (ImGui::MenuItem("Copy", "CTRL+C")) {}

// if (ImGui::MenuItem("Paste", "CTRL+V")) {}

ImGui::EndMenu();

}

ImGui::EndMainMenuBar();

}

if (isReset)

{

ImGui::Begin("ResetWindow", &isReset);

ImGui::Text("Reset?");

if (ImGui::Button("Yes"))

{

g\_pObjectManager->Destroy();

isReset = false;

}

ImGui::SameLine();

if (ImGui::Button("No"))

isReset = false;

ImGui::End();

}

} // << : Menu Title Bar

{ // >> : Controller

// >> 렉걸려서 주석처리

// static LPDIRECT3DTEXTURE9 temp;

// D3DXCreateTextureFromFileA(g\_pD3DDevice, "Resource/Test.png", &temp);

// static ImTextureID test = temp;

ImGui::Begin(" ");

/\*for (int i = 0; i < 5 ; i++)

{

ImGui::PushID(i);

int frame\_padding = 2;

ImVec2 size = ImVec2(32.0f, 32.0f);

ImVec2 uv0 = ImVec2(0.0f, 0.0f);

ImVec2 uv1 = ImVec2(32.0f / 32, 32.0f / 32);

ImVec4 bg\_col = ImVec4(0.0f, 0.0f, 0.0f, 1.0f);

ImVec4 tint\_col = ImVec4(1.0f, 1.0f, 1.0f, 1.0f);

if (ImGui::ImageButton(test, size, uv0, uv1, frame\_padding, bg\_col, tint\_col))

{

if (i == 0) cout << "Mouse Move Mode" << endl;

else if (i == 1) cout << "Object Move Mode" << endl;

else if (i == 2) cout << "Object Rotate Mode" << endl;

else if (i == 3) cout << "Object Scale Mode" << endl;

else if (i == 4) cout << "Object OBB Mode" << endl;

}

ImGui::SameLine();

ImGui::PopID();

}\*/

ImGui::End();

} // << : Controller

{

ImGui::Begin("Hiearachy");

static bool temp = false;

static bool temp2 = false;

for (int i = 0; i < g\_pObjectManager->GetVecObject().size(); i++)

{

bool isClick = g\_pObjectManager->GetObejctClick(i);

if (ImGui::Selectable(g\_pObjectManager->GetOjbectName(i).c\_str(), &isClick))

{

g\_pObjectManager->SetObjectClick(i, isClick);

cout << "click" << endl;

}

}

// !! 주의 !! : 이름 같으면 안됨!

ImGui::Selectable("todo Something", &temp);

ImGui::Selectable("todo Something2", &temp2);

/\*

if (ImGui::TreeNode("Basic"))

{

static bool selection[5] = { false, true, false, false, false };

ImGui::Selectable("1. I am selectable", &selection[0]);

ImGui::Selectable("2. I am selectable", &selection[1]);

ImGui::Text("3. I am not selectable");

ImGui::Selectable("4. I am selectable", &selection[3]);

if (ImGui::Selectable("5. I am double clickable", selection[4], ImGuiSelectableFlags\_AllowDoubleClick))

if (ImGui::IsMouseDoubleClicked(0))

selection[4] = !selection[4];

ImGui::TreePop();

}

if (ImGui::TreeNode("Selection State: Single Selection"))

{

static int selected = -1;

for (int n = 0; n < 5; n++)

{

char buf[32];

sprintf(buf, "Object %d", n);

if (ImGui::Selectable(buf, selected == n))

selected = n;

}

ImGui::TreePop();

}

if (ImGui::TreeNode("Selection State: Multiple Selection"))

{

HelpMarker("Hold CTRL and click to select multiple items.");

static bool selection[5] = { false, false, false, false, false };

for (int n = 0; n < 5; n++)

{

char buf[32];

sprintf(buf, "Object %d", n);

if (ImGui::Selectable(buf, selection[n]))

{

if (!ImGui::GetIO().KeyCtrl) // Clear selection when CTRL is not held

memset(selection, 0, sizeof(selection));

selection[n] ^= 1;

}

}

\*/

ImGui::End();

}

{ // >> : FileLoad

ImGui::Begin("File Loader");

static int index = -1;

enum class LoadType { eMap, eBackground, eObject };

static LoadType mode = LoadType::eBackground;

if (ImGui::RadioButton("Background", mode == LoadType::eBackground)) { mode = LoadType::eBackground; index = -1; }

ImGui::SameLine();

if (ImGui::RadioButton("Object", mode == LoadType::eObject)) { mode = LoadType::eObject; index = -1; }

vector<char \*> vecItem;

if (mode == LoadType::eBackground)

{

vecItem.push\_back("BackObj"); vecItem.push\_back("BackObj1");

// >> todo : item 추가

}

else if (mode == LoadType::eObject)

{

vecItem.push\_back("Obj"); vecItem.push\_back("Obj1");

// >> todo : item 추가

}

// << combo

const char\* showItem;

if (index == -1)

showItem = " ";

else

showItem = vecItem[index];

if (ImGui::BeginCombo(" ", showItem))

{

for (int n = 0; n < vecItem.size(); n++)

{

const bool is\_selected = (index == n);

if (ImGui::Selectable(vecItem[n], is\_selected))

index = n;

if (is\_selected)

ImGui::SetItemDefaultFocus(); // focus

}

ImGui::EndCombo();

}

// << combo

ImGui::SameLine();

if (ImGui::Button("Load") && index != -1)

{

// >> todo : File load

cout << "load" << endl;

}

ImGui::Separator();

ImGui::End();

} // << : FileLoad

{

ImGui::Begin("Inspector");

ImGui::End();

}