第三回(課題２)クライアント側コード

```
#include "02ClientAssignment.h"

//-定数----------------------------------------------

#define BUFFER_SIZE         128
#define XYZ000     DirectX::XMFLOAT3(0.0f,0.0f,0.0f)
#define PLAYER_SIZE DirectX::XMFLOAT3(0.02f,0.02f,0.02f)
#define MESS_WIND_SIZE_X    250
#define MESS_WIND_SIZE_Y    400
#define COM_WIND_SIZE_X     420
#define COM_WIND_SIZE_Y             650
#define  COM_WIND_POS_X             50
#define COM_WIND_POS_Y              30
#define PORT                        7000
#define PORTSTR                     "7000"
//-------コンストラクタ-------------------------------------------------
ClientAssignment02::ClientAssignment02()
{
         input[0] = '¥0';
}

//-------デストラクタ---------------------------------------------------
ClientAssignment02::~ClientAssignment02()
{
}

//-------初期化メソッド-------------------------------------------------
void ClientAssignment02::Initialize()
{
         //初期化メソッドごと前回作成したもののコピー＆ペーストでOKですが、
         //最後のスレッド立ち上げは
         //th = std::thread(&ClientAssignment02::recvThread, this);
         //なので注意


         SceneBase::Initialize();
```

第三回(課題２)クライアント側コード

```
	// TODO 01_10
	// WinsockAPI を初期化
	if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
	{
		// 初期化失敗
		Logger::Print("WSA Initialize Failed.¥n");
		return;
	}

	// TODO 01_11
	// DNS へ問い合わせ
	addrinfo hints;                                    // DNS へ問い合わせに必要な情報を
設定
	addrinfo* addrInfo = NULL;  // 取得したアドレスがここに保存される
	// ゼロクリア
	ZeroMemory(&hints, sizeof(addrinfo));
	// 設定する情報を設定
	hints.ai_family = AF_INET;                         // IPv4 で取得
	hints.ai_socktype = SOCK_STREAM;     // TCP 通信で DNS サーバへアクセス

	const char hostname[] = "localhost"; // ドメイン指定
	const char port[] = "7000";          // ポート番号指定
	// DNS へ問い合わせ 0 のとき正常に完了
	if (getaddrinfo(hostname, port, &hints, &addrInfo) != 0) {
		Logger::Print("getaddrinfo error.¥n");
		return;
	}

	// TODO 01_12
	// WinsockAPI を初期化
	// 取得した IP アドレスを sockaddr_in に変換し sin_addr を代入
	addr.sin_addr = reinterpret_cast<sockaddr_in*>(addrInfo->ai_addr)->sin_addr;
	addr.sin_family = AF_INET;
	addr.sin_port = htons(PORT);
```

第三回(課題２)クライアント側コード

```
	// TODO 01_13
	// ソケット作成
	sock = socket(AF_INET, SOCK_DGRAM, 0);
	if (sock == INVALID_SOCKET) {
		Logger::Print("create socket Failed.¥n");
		return;
	}

	// TODO 01_14
	// ノンブロッキング設定
	u_long mode = 1;
	int m = ioctlsocket(sock, FIONBIO, &mode);
	if (m != 0)
	{
		Logger::Print("Nonblocking Mode Failed.¥n");
		return;
	}

	// TODO 01_15
	// クライアント情報をサーバへ知らせるために sendto 関数を最初に実行する。
	int size = sendto(sock, "connect", 8, 0, reinterpret_cast<sockaddr*>(&addr),
sizeof(addr));
	if (size < 0)
	{
		Logger::Print("connected failed. error code:%d¥n", WSAGetLastError());
	}

	Logger::Print("Connect Success¥n");

	// Player 設定
	playerManager = new PlayerManager();
	playerManager->SetMyPlayerID(0);

	Player* player = new Player();
	// 各プレイヤーのキャラクターに合わせた設定を行う
	// JobClass はステートマシン
```

第三回(課題２)クライアント側コード

```
	player->SetPlayerID(0);
	player->SetPosition(XYZ000);
	player->SetScale(PLAYER_SIZE);
	player->SetState(Player::State::Idle);
	player->SetModel();
	player->SetPlayerID(0);

	playerManager->AddPlayer(player);
	playerManager->GetMyPlayer()->SetReady(true);

	// 受信スレッド実装
	th = std::thread(&ClientAssignment02::recvThread, this);
}

//-------終了メソッド-----------------------------------------------
void ClientAssignment02::Finalize()
{
	//終了メソッドも前回作成したもののコピー＆ペーストでOKです。

	// マルチスレッドのループフラグを下ろす
	loop = false;

	// スレッドの終了まで待機
	th.join();

	// TODO 01_18
	// ソケット終了
	if (closesocket(sock) != 0) {
		int err = WSAGetLastError();
		Logger::Print("Close Socket Failed.error_code:%d¥n", err);
		{
		}
	}
```

第三回(課題２)クライアント側コード

```
	// TODO 01_19
	// WSA 終了処理
	if (WSACleanup() != 0)
	{
		int err = WSAGetLastError();
		Logger::Print("Cleanup WinsockAPI Failed.%d¥n", err);
	}
}

//-------描画メソッド-----------------------------------------------
void ClientAssignment02::Render()
{
	// 基底クラスのレンダー呼び出し
	SceneBase::Render();
	// imgui
	ImGui::SetNextWindowPos(ImVec2(50, 50), ImGuiCond_Once);
	ImGui::SetNextWindowSize(ImVec2(300, 100), ImGuiCond_Once);
	if (ImGui::Begin("Title", nullptr, ImGuiWindowFlags_None))
	{
		if (ImGui::Button(u8"タイトルへ"))
		{
			SceneManager::Instance().ChangeScene(new SceneTitle());
		}
	}
	ImGui::End();
}

//-------更新処理メソッド-------------------------------------------
void ClientAssignment02::NetrowkUpdate(float elapsedTime)
{
	// 操作キャラクター取得
	Player* player = playerManager->GetMyPlayer();

	// カメラコントローラー更新処理
	if (player != nullptr)
	{
```

第三回(課題２)クライアント側コード

```
		HWND hWnd;
		hWnd = GetActiveWindow();
		if (hWnd != NULL)
		{
			// キー入力判定(入力データはそのままサーバに送らず更新)
			switch (player->GetState())
			{
			case Player::State::Idle:
				if (player->InputMove(elapsedTime))
				{
					if	(player)player->GetJobClass()-
>ChangeState(Player::State::Move);
				}
				if (player->InputAttack())
				{
					player->GetJobClass()-
>ChangeState(Player::State::Attack);
				}
				break;
			case Player::State::Move:
				if (!player->InputMove(elapsedTime))
				{
					player->GetJobClass()-
>ChangeState(Player::State::Idle);
				}
				if (player->InputAttack())
				{
					player->GetJobClass()-
>ChangeState(Player::State::Attack);
				}
				break;
			case Player::State::Attack:
				if (!player->GetModel()->IsPlayAnimation())
				{
					player->GetJobClass()-
>ChangeState(Player::State::Idle);
```

第三回(課題２)クライアント側コード

```
                                        }
                                        break;
                                }
                        }
                }

        PlayerInformation plInfo{};
        // 送信データ
        plInfo.position = player->GetPosition();
        plInfo.angle = player->GetAngle();
        plInfo.state = player->GetState();

        int size = 0;
        //今回新規作成部分 C01
        // TODO 02_10
        // plInfo をサーバに送信する

        size = sendto(sock, reinterpret_cast<char*>(&plInfo), sizeof(PlayerInformation),
        0, reinterpret_cast<sockaddr*>(&addr), static_cast<int>(sizeof(sockaddr_in)));

        if (size < 0)
        {
                Logger::Print("sendto failed. error code:%d¥n", WSAGetLastError());
        }
}

//-------サーバーからの受信スレッド-----------------------------------------------
void ClientAssignment02::recvThread()
{
        int len = sizeof(sockaddr_in);
        do {
                PlayerInformation plInfo{};
```

第三回(課題２)クライアント側コード

```
//今回新規作成部分 C02
// TODO 02_11
// データ受信を行い、
//プレイヤー情報を受信する
// 受信データは plInfo に保存だけでプレイヤーには反映しない
```

```
int size = recvfrom(sock, reinterpret_cast<char*>(&plInfo), sizeof(PlayerInformation), 0, reinterpret_cast<sockaddr*>(&recvAddr), &len);
```

```
if (size < 0)
{
	if (WSAGetLastError() != WSAEWOULDBLOCK)
	{
		Logger::Print("recvfrom failed. error code:%d¥n", WSAGetLastError());
	}
}


	} while (loop);
}
```