포트폴리오

Client Software Engineer

오정훈

ojh650765@tukorea.ac.kr

https://github.com/ojh6507

목차

- Unreal5 비대칭 PVP 술래잡기, “The Toys” (C++20)
- Android Studio 로그라이트, “Pixel Game” (Java)
- Pico2d, Super Mario (Python)

Unreal5 비대칭 PVP 술래잡기 (C++20)

"The Toys" 2023/08 ~ 2024/07, 3인 팀, 클라이언트 프로그래머로 참여

캐릭터 스킬로 술래를 피해 퓨즈를 모아
포탈을 활성화한 뒤 탈출하는 술래잡기 게임

한국공학대학교 졸업 작품

Unreal5 비대칭 PVP 술래잡기 (C++20)

“The Toys”

팀

- 팀 구성: 3인 팀 (클라이언트: 1, 서버: 1, 그래픽/기획: 1)
- 팀 내 역할 : 메인 클라이언트 프로그래머

주요 작업 (본인 파트)

- 네트워크 연결 및 패킷 재조립을 제외한 클라이언트 코드 전반 개발
- 큐에 전달된 패킷을 게임 내에서 처리
- 젤리 폭발 이펙트 구현
- 스킬 이펙트 구현

클라이언트 소스코드 깃 허브:

https://github.com/NewbieProgrammerCrew/graduation-project/tree/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World

(Public 폴더에 헤더파일, Private 폴더에 cpp파일 있습니다)

팀 작업 일지:

https://github.com/NewbieProgrammerCrew/graduation-project/tree/main/%EC%9E%91%EC%97%85%EC%9D%BC%EC%A7%80

유튜브:

https://youtu.be/FtYJOIR3v90

Unreal5 비대칭 PVP 술래잡기 (C++20)

"The Toys" 코드

도망자 폭탄 발사 코드 입니다.
CurrentBombType을 사용하여 폭탄의 종류를 관리하고,
ShootCannon 함수에서 폭탄을 선택적으로 생성 및 발사합니다.

```cpp
void ABaseRunner::ShootCannon(FVector pos, FVector dir)
{
    if (CurrentBombType != BombType::NoBomb) {
        PlayMontage(BombMontage, "Shoot");
        UClass* BP_StunBombClass = nullptr;
        UClass* BP_ExplosiveBombClass = nullptr;
        UClass* BP_InkBombClass = nullptr;
        ABomb* bomb = nullptr;
        switch (CurrentBombType)
        {
        case BombType::Explosion: {
            BP_ExplosiveBombClass = LoadClass<ABomb>(nullptr,
                                    TEXT("Blueprint'/Game/Blueprints/MyActor/BP_ExplosiveBomb.BP_ExplosiveBomb_C'"));
            bomb = GetWorld()->SpawnActor<ABomb>(BP_ExplosiveBombClass);
            bomb->SetActorLocation(BombShootArrowComponent->GetComponentLocation());
            break;
        }
        case BombType::Stun: {
            BP_StunBombClass = LoadClass<ABomb>(nullptr,
                               TEXT("Blueprint'/Game/Blueprints/MyActor/BP_StunBomb.BP_StunBomb_C'"));
            bomb = GetWorld()->SpawnActor<ABomb>(BP_StunBombClass);
            bomb->SetActorLocation(BombShootArrowComponent->GetComponentLocation());
            break;
        }
        case BombType::Blind: {
            BP_InkBombClass = LoadClass<ABomb>(nullptr,
                              TEXT("Blueprint'/Game/Blueprints/MyActor/BP_InkBomb.BP_InkBomb_C'"));
            bomb = GetWorld()->SpawnActor<ABomb>(BP_InkBombClass);
            bomb->SetActorLocation(BombShootArrowComponent->GetComponentLocation());
            break;
        }
        default:
            bomb = nullptr;
            break;
        }
        if (IsValid(bomb)) {
            bomb->SetType(CurrentBombType);
            bomb->Fire(pos, dir, 50);
            CurrentBombType = BombType::NoBomb;
        }

    }
}
```

BaseRunner코드 전체:
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/Actors/BaseRunner.cpp

Unreal5 비대칭 PVP 술래잡기 (C++20)

“The Toys” 코드

도망자와 오브젝트 상호작용 코드입니다.

FindItemBoxAndCheckEquipableBomb 함수는 라인 트레이스를 사용해 플레이어가 아이템 상자 또는 퓨즈 박스와 상호작용할 수 있는지 여부를 확인하고, 상호작용 UI를 동적으로 업데이트 합니다.

```cpp
bool ABaseRunner::FindItemBoxAndCheckEquipableBomb(FVector CameraLocation, FRotator CameraRotation, float distance)
{
    FHitResult Hit = PerformLineTrace(CameraLocation, CameraRotation, distance);
    AItemBox* HitItemBox = Cast<AItemBox>(Hit.GetActor());
    AFuseBox* HitFuseBox = Cast<AFuseBox>(Hit.GetActor());

    if (HitItemBox) {
        SetCurrentItemBox(HitItemBox);
        prevItemBox = HitItemBox;
        bool boxOpened;
        HitItemBox->GetBoxStatus(boxOpened);
        if (boxOpened) {
            ProcessCustomEvent(HitItemBox, FName("DisavailableItemBox"));
            ProcessEvent(HideBoxOpeningUIEvent, nullptr);
            ClearOpeningBoxData();
        }
        else {
            ProcessCustomEvent(HitItemBox, FName("AvailableItemBox"));
            ProcessEvent(ShowBoxOpeningUIEvent, nullptr);
        }
        checkItemBoxAvailable();
        if (UpdateEquipableBombData(Hit, HitItemBox)) {
            ProcessEvent(ShowBombAcquiredUIEvent, nullptr);
            return true;
        }
        else {
            ProcessEvent(HideBombAcquiredUIEvent, nullptr);
            return false;
        }
    }
    else if (HitFuseBox) {
        ClearOpeningBoxData();
        return FindFuseBoxInViewAndCheckPutFuse(HitFuseBox);

    }
    else {
        ProcessCustomEvent(FuseBox, FName("ChangeInvalidColor"));
        ProcessCustomEvent(prevItemBox, FName("DisavailableItemBox"));
        prevItemBox = nullptr;

        ProcessEvent(HideBombAcquiredUIEvent, nullptr);
        ProcessEvent(HideBoxOpeningUIEvent, nullptr);
        ProcessEvent(HideUIEvent, nullptr);
        ClearOpeningBoxData();
        SetFuseBoxOpenAndInstall(-1);
        //SetOpeningFuseBox(false);
        return false;
    }
}
```

초기 프로젝트 설계 시, 빠른 프로토타이핑과 개발 속도를 고려하여 Blueprint 이벤트를 사용해 상호작용을 처리했습니다.

그러나, 개발 과정에서 Blueprint 호출이 많아지면서 성능에 영향을 미칠 수 있다는 점을 인식했습니다.
이를 개선하기 위해 **현재 C++ 함수로 전환하는 작업을 진행하고 있습니다**.

BaseRunner코드 전체:
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/Actors/BaseRunner.cpp

Unreal5 비대칭 PVP 술래잡기 (C++20)

“The Toys” 코드

젤리 오브젝트 매니저 코드입니다.

InitJelly함수에서 level에 존재하는 Jelly 오브젝트들을 찾고 정렬하여 인덱스를 부여합니다.

젤리 폭발 이펙트 (버텍스 애니메이션) 재생

서버에서 부딪힌 Bomb 위치를 전달하면 Bomb과 젤리가 서로 마주보도록 했습니다. (자연스러운 폭발 이펙트를 위해)

네트워크를 관리하는 스레드에서 호출되므로 SetActorRotation 함수는 게임 스레드에서 처리하도록 구현했습니다.

```cpp
void AJellyManager::InitJelly()
{
    UGameplayStatics::GetAllActorsOfClass(GetWorld(), JellyActor->GetClass(), jellies);
    jellies.Sort([&](const AActor& A, const AActor& B) {
        return A.GetName() < B.GetName();
        });
    int idx{};
    for (auto j : jellies) {
        Cast<AJelly>(j)->SetIndex(idx);
        ++idx;
    }
}

void AJellyManager::ExplosionParticleEvent(int idx)
{
    if (jellies[idx]) {
        Cast<AJelly>(jellies[idx])->ExplosionEffect();
    }
}

void AJellyManager::LookAtBomb(FVector bombLocation, int idx)
{
    AsyncTask(ENamedThreads::GameThread, [this, idx,bombLocation]() {
        if (!IsValid(jellies[idx])) return; // 유효성 검사 추가

        FVector CurrentLocation = jellies[idx]->GetActorLocation();
        FRotator CurrentRot = GetActorRotation();
        FRotator TargetRot = UKismetMathLibrary::FindLookAtRotation(CurrentLocation, bombLocation);
        FRotator NewRot = FRotator(CurrentRot.Pitch, TargetRot.Yaw + 180.f, CurrentRot.Roll);

        jellies[idx]->SetActorRotation(NewRot);
        });
}
```

JellyManager코드 전체:
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/Manager/JellyManager.cpp

Unreal5 비대칭 PVP 술래잡기 (C++20)

"The Toys" 코드

PlayerManager코드 전체:
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/Manager/PlayerManager.cpp

Player Manager.h

Player Manager코드입니다.

PlayerManager.cpp에서는 매 tick마다 서버로부터 받은 패킷을 try_pop으로 concurrent_queue에서 꺼내어 게임 내 여러 이벤트와 상호작용을 처리합니다.

Ex) 플레이어 캐릭터 spawn과 possess처리

```cpp
void APlayerManager::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    if (Network == nullptr && Main->init_finish) {
        Network = reinterpret_cast<FSocketThread*>(Main->Network);
        if (!Network) return;
        Network->_PlayerManager = this;
        lobby_id = Main->GameInstance->GetMyLobbyID();
        game_id = Main->GameInstance->GetMyGameID();
    }
    SC_ADD_PLAYER_PACKET AddPlayer;
    while (!PlayerQueue.empty()) {
        if (PlayerQueue.try_pop(AddPlayer)) {
            Spawn_Player(AddPlayer);
        }
    }
    SC_MOVE_PLAYER_PACKET move_player;
    while (!Player_Move_Queue.empty()) {
        if (Player_Move_Queue.try_pop(move_player)) {
            FRotator Rotation = FRotator(move_player.ry, move_player.rz, move_player.rx);
            FVector location = FVector(move_player.x, move_player.y, move_player.z);
            double pitch = move_player.pitch;
            cur_speed = move_player.speed;
            cur_jump = move_player.jump;
            Set_Player_Location(move_player.id, location, Rotation, pitch);
        }
    }

    SC_ATTACK_PLAYER_PACKET attack_player;
    while (!Player_Attack_Queue.empty()) {
        if (Player_Attack_Queue.try_pop(attack_player)) {
            Play_Attack_Animation(attack_player);
        }
    }
    SC_HITTED_PACKET hitted_player;
    while (!Player_Hitted_Queue.empty()) {
        if (Player_Hitted_Queue.try_pop(hitted_player)) {
            Player_Hitted(hitted_player);
        }
    }
    SC_DEAD_PACKET dead_player;
    while (!Player_Dead_Queue.empty()) {
        if (Player_Dead_Queue.try_pop(dead_player)) {
            Player_Dead(dead_player);
        }
    }
    SC_CHASER_RESURRECTION_PACKET resurrection_chaser;
    while (!Player_Resurrection_Queue.empty()) {
        if (Player_Resurrection_Queue.try_pop(resurrection_chaser)) {
            Player_Resurrect(resurrection_chaser);
        }
    }
    SC_OPENING_ITEM_BOX_PACKET item_Box_OpeningPlayer;
    while (!Player_Opening_ItemBox_Queue.empty()) {
```

```cpp
void APlayerManager::Spawn_Player(SC_ADD_PLAYER_PACKET AddPlayer) {
    if (!(Main && Main->GameInstance))return;
    int mapid = Main->GameInstance->GetMapId() - 1;
    UWorld* uworld = GetWorld();
    if (!uworld) return;

    if (!(std::string(AddPlayer.role).size() && AddPlayer.id >= 0)) return;

    ACharacter* SpawnedCharacter = nullptr;
    int characterTypeNumer = AddPlayer.charactorNum;


    if (Player[AddPlayer.id] != nullptr) {
        UpdateCharacterPosition(AddPlayer.id, PositionArray[mapid][AddPlayer.id % 5]);
    }
    else {
        SpawnNewCharacter(uworld, AddPlayer, characterTypeNumer, PositionArray[mapid][AddPlayer.id %5]);
    }
    AsyncTask(ENamedThreads::GameThread, [uworld]() {
        ALevelScriptActor* LevelScriptActor = uworld->GetLevelScriptActor();
        if (LevelScriptActor) {
            UFunction* Function = LevelScriptActor->FindFunction(FName("DestroyLoadWidget"));
            if (Function) {
                LevelScriptActor->ProcessEvent(Function, nullptr);
            }
        }
    });
}

void APlayerManager::SpawnNewCharacter(UWorld* uworld, SC_ADD_PLAYER_PACKET& AddPlayer, int characterTypeNumer, FVector pos)
{
    if (!PlayerBPMap.Contains(characterTypeNumer)) return;

    FVector SpawnLocation(pos);
    ACharacter* SpawnedCharacter = uworld->SpawnActor<ACharacter>(PlayerBPMap[characterTypeNumer], SpawnLocation, FRotator::ZeroRotator);
    if (!SpawnedCharacter) return;
    Player[AddPlayer.id] = Cast<AActor>(SpawnedCharacter);
    if (Player[AddPlayer.id]) {
        if (!PossessCharacter(AddPlayer.id)) {
            Main->SendMapLoadedPacket();
        }
        else {
            UpdateCharacterData(AddPlayer, characterTypeNumer);
            ABaseRunner* runner = Cast<ABaseRunner>(SpawnedCharacter);
            if (runner) {
                runner->SetRunnerType(characterTypeNumer);
            }
            ABaseChaser* chaser = Cast<ABaseChaser>(SpawnedCharacter);
            if (chaser) {
                chaser->SetChaserType(characterTypeNumer - 5);
            }
        }
    }
}

bool APlayerManager::PossessCharacter(int playerId)
{
    if (Network && playerId == Network->my_game_id) {
        APlayerController* RawController = UGameplayStatics::GetPlayerController(this, 0);
        ACh_PlayerController* MyController = Cast<ACh_PlayerController>(RawController);
        if (MyController) {
            MyController->Possess(Cast<APawn>(Player[playerId]));
            return true;
        }
    }
    return false;
}

void APlayerManager::UpdateCharacterData(SC_ADD_PLAYER_PACKET& AddPlayer, int characterTypeNumer)
{
    ABaseCharacter* baseCharacter = Cast<ABaseCharacter>(Player[AddPlayer.id]);
    if (baseCharacter) {
        baseCharacter->SetRole(FString(AddPlayer.role));
        baseCharacter->SetHPData(AddPlayer._hp);
        baseCharacter->SetCharacterType(characterTypeNumer);
    }
}

void APlayerManager::UpdateCharacterPosition(int playerId, FVector position)
{
    Player[playerId]->SetActorHiddenInGame(false);
    Player[playerId]->SetActorLocation(position);
}
```

```cpp
void SetMyLobbyID(int id) { lobby_id = id; }
void SetMyGameID(int id) { game_id = id; }
void SetPlayerQueue(SC_ADD_PLAYER_PACKET* packet);
void Set_Player_Move_Queue(SC_MOVE_PLAYER_PACKET* MovePacket);
void Set_Player_Attack_Queue(SC_ATTACK_PLAYER_PACKET* AttackPacket);
void Set_Player_Hitted_Queue(SC_HITTED_PACKET* HittedPacket);
void Set_Player_Dead_Queue(SC_DEAD_PACKET* DEADPacket);
void Set_Player_Escape_Queue(SC_ESCAPE_PACKET* Packet);
void Set_Chaser_Win_Queue(SC_CHASER_WIN_PACKET* Packet);
void Set_Player_Resurrect_Queue(SC_CHASER_RESURRECTION_PACKET* ResurrectPacket);
void Set_Player_Fuse_Pickup_Queue(SC_PICKUP_FUSE_PACKET * PickupPacket);
void Set_Player_Bomb_Pickup_Queue(SC_PICKUP_BOMB_PACKET* PickupPacket);

void Set_Player_Explosion_Pickup_Queue(SC_PICK_UP_EXPLOSION_PACKET* packet);
void Set_Player_Ink_Pickup_Queue(SC_PICK_UP_INK_PACKET* packet);
void Set_Player_Stun_Pickup_Queue(SC_PICK_UP_STUN_PACKET* packet);

void Set_Player_Remove_Queue(SC_REMOVE_PLAYER_PACKET* RemovePacket);
void Set_Player_Aiming_Queue(SC_AIM_STATE_PACKET* AimPacket);
void Set_Player_FireCannon_Queue(SC_CANNON_FIRE_PACKET* FirePacket);
void Set_Player_Idle_Queue(SC_IDLE_STATE_PACKET* IdlePacket);
void Set_Player_Use_Skill_Queue(SC_USE_SKILL_PACKET* Packet);
void Set_Student_Player_Choosed_Skill_Queue(SC_SKILL_CHOOSED_PACKET* Packet);
void Set_Player_ItemBoxOpening_Queue(SC_OPENING_ITEM_BOX_PACKET* ItemOpeningPacket);
void Set_Player_FuseBoxOpening_Queue(SC_OPENING_FUSE_BOX_PACKET* packet);
void Set_Player_Stop_Opening_Queue(SC_STOP_OPENING_PACKET* packet);

void Choosed_Skill_Student_Player(SC_SKILL_CHOOSED_PACKET packet);
void Spawn_Player(SC_ADD_PLAYER_PACKET packet);
void SpawnNewCharacter(UWorld* uworld, SC_ADD_PLAYER_PACKET& AddPlayer, int characterTypeNumer, FVector Pos);
bool PossessCharacter(int playerId);
void UpdateCharacterData(SC_ADD_PLAYER_PACKET& AddPlayer, int characterTypeNumer);
void UpdateCharacterPosition(int playerId, FVector position);
void Set_Player_Location(int citizen_id, FVector Packet_Location, FRotator Rotate, double pitch);
void Player_Escape(SC_ESCAPE_PACKET packet);
void Chaser_Win(SC_CHASER_WIN_PACKET packet);

void Play_Attack_Animation(SC_ATTACK_PLAYER_PACKET packet);
void Player_Hitted(SC_HITTED_PACKET hitted_player);
void Player_FUSE_Pickup(SC_PICKUP_FUSE_PACKET item_pickup_player);
void PortalGagueUpdate(float ratio);
void Player_Bomb_Pickup(SC_PICKUP_BOMB_PACKET item_pickup_player);

//debug
void Player_Explosion_Pickup(SC_PICK_UP_EXPLOSION_PACKET packet);
void Player_Stun_Pickup(SC_PICK_UP_STUN_PACKET packet);
void Player_Ink_Pickup(SC_PICK_UP_INK_PACKET packet);


void Player_Fire_Cannon(SC_CANNON_FIRE_PACKET fireCannonPlayer);
void Play_Aim_Animation(SC_AIM_STATE_PACKET aim_player);
void Play_Idle_Animation(SC_IDLE_STATE_PACKET idle_player);
void Player_Use_Skill(SC_USE_SKILL_PACKET skill_player);
void Player_Opening_ItemBox(SC_OPENING_ITEM_BOX_PACKET packet);
void Player_Opening_FuseBox(SC_OPENING_FUSE_BOX_PACKET packet);
void Player_Stop_Opening_Box(SC_STOP_OPENING_PACKET packet);
//void Player_Reset_FuseBox(SC_RESET_FUSE_BOX_PACKET packet);

void Player_Dead(SC_DEAD_PACKET dead_player);
void Player_Resurrect(SC_CHASER_RESURRECTION_PACKET player);
void Remove_Player(int _id);


ivate:
concurrency::concurrent_queue <SC_ADD_PLAYER_PACKET> PlayerQueue;
concurrency::concurrent_queue <SC_MOVE_PLAYER_PACKET> Player_Move_Queue;
concurrency::concurrent_queue<SC_ATTACK_PLAYER_PACKET> Player_Attack_Queue;
concurrency::concurrent_queue<SC_HITTED_PACKET> Player_Hitted_Queue;
concurrency::concurrent_queue <SC_DEAD_PACKET> Player_Dead_Queue;
concurrency::concurrent_queue <SC_PICKUP_FUSE_PACKET> Player_Fuse_Pickup_Queue;
concurrency::concurrent_queue <SC_PICKUP_BOMB_PACKET> Player_Bomb_Pickup_Queue;

concurrency::concurrent_queue <SC_PICK_UP_INK_PACKET> Player_Ink_Pickup_Queue;
concurrency::concurrent_queue <SC_PICK_UP_EXPLOSION_PACKET> Player_Explosion_Pickup_Queue;
concurrency::concurrent_queue <SC_PICK_UP_STUN_PACKET> Player_Stun_Pickup_Queue;

concurrency::concurrent_queue <SC_REMOVE_PLAYER_PACKET> Player_Remove_Queue;
concurrency::concurrent_queue <SC_ESCAPE_PACKET> Player_Escape_Queue;
concurrency::concurrent_queue <SC_CHASER_WIN_PACKET> Chaser_Win_Queue;

concurrency::concurrent_queue <SC_AIM_STATE_PACKET> Player_Aim_Queue;
concurrency::concurrent_queue <SC_IDLE_STATE_PACKET> Player_Idle_Queue;
concurrency::concurrent_queue <SC_USE_SKILL_PACKET> Player_Use_Skill_Queue;
concurrency::concurrent_queue <SC_OPENING_ITEM_BOX_PACKET> Player_Opening_ItemBox_Queue;
concurrency::concurrent_queue <SC_OPENING_FUSE_BOX_PACKET> Player_Opening_FuseBox_Queue;
//concurrency::concurrent_queue <SC_RESET_FUSE_BOX_PACKET> Player_Reset_FuseBox_Queue;
concurrency::concurrent_queue <SC_STOP_OPENING_PACKET> Player_Stop_Opening_Queue;
concurrency::concurrent_queue <SC_CANNON_FIRE_PACKET> Player_Fire_Cannon_Queue;
concurrency::concurrent_queue <SC_CHASER_RESURRECTION_PACKET> Player_Resurrection_Queue;
concurrency::concurrent_queue <SC_SKILL_CHOOSED_PACKET> Student_Choosed_Skill_Queue;

int game_id{ -1 };
```

Unreal5 비대칭 PVP 술래잡기 (C++20)

“The Toys” 젤리 폭발 구현

접근 방법 1: Unreal Engine의 Fluid Simulation과 Chaos Destruction 사용

- 첫 번째 접근 방법은 Unreal Engine의 Fluid Simulation과 Chaos Destruction 시스템을 활용하여 젤리 폭발 이펙트를 구현하는 것이었습니다.
- 하지만, 이 방법은 연성체(soft body) 폭발이 아닌 강성체(rigid body) 폭발에 더 적합하여 시각적으로 기대한 결과를 얻지 못했습니다.

접근 방법 2: CUDA와 SPH(입자 기반 유체) 기법 활용

- 두 번째 접근 방법은 CUDA 라이브러리를 사용하여 SPH(입자 기반 유체) 기법을 적용하는 것이었습니다.
- 이 라이브러리는 Unreal Engine과 연동되고 파티클의 움직임을 실시간으로 계산하는데 사용했습니다.
- CUDA를 통해 파티클의 위치와 움직임을 계산한 후, Unreal Engine 액터로 정보를 전달하고, 이를 다시 GPU로 전송해 렌더링하는 과정에서 성능 비효율이 발생했습니다.
- 이러한 비효율성으로 인해 젤리 하나를 처리하는 데 fps는 40fps 이하로 떨어졌습니다.
- Cuda와 상호작용 하는 액터 소스코드: https://github.com/ojh6507/Cuda_UE5/blob/main/Source/CUDATest/MyActor.cpp

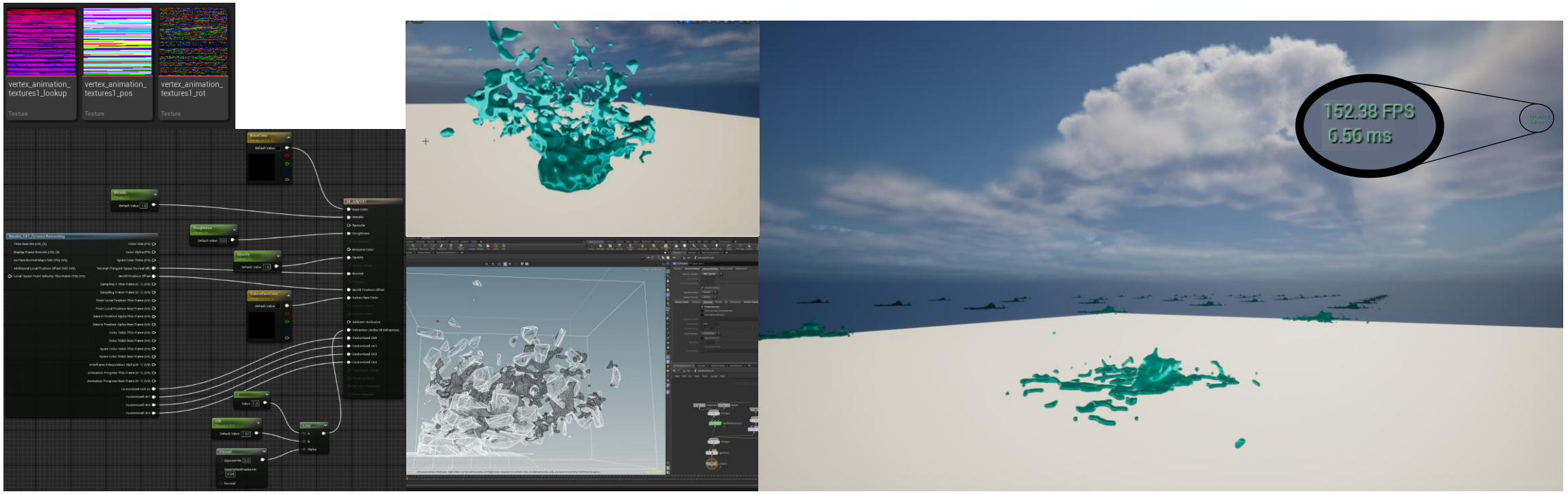
Unreal5 비대칭 PVP 슬래잡기 (C++20)

“The Toys” 젤리 폭발 구현

접근 방법 3: Vertex Animation Texture (VAT) 활용

- 세 번째 접근 방법은 Vertex Animation Texture (VAT)를 사용해 GPU에서 모든 연산을 처리하는 방식이었습니다.
- 실시간 연산 대신 미리 연산된 데이터를 활용하면, 프레임 속도를 크게 향상시킬 수 있을 것이라 판단했습니다.
- 실제로 VAT를 사용한 젤리 30개 폭발 이펙트 재생 시 140 ~ 160 fps로 향상되었습니다.
- 따라서, 세 번째 접근 방법인 Vertex Animation Texture (VAT)로 프로젝트를 진행했습니다.

VAT (hdr)



Android Studio, 로그라이트 (Java)

"Pixel Game" 2024/03 ~ 2024/06, 1인 개발, 클라이언트 프로그래머로 참여

턴마다 각도를 계산해 폭탄을 피하고 아이템을 모아
슬라임을 공격하는 로그라이트 퍼즐 게임

스마트폰 게임 프로그래밍 텀 프로젝트

Android Studio, 로그라이트 (Java)

“Pixel Game”

주요 작업 (본인 파트)

- 모든 클라이언트 코드 제작
- Sprite 에셋 제작

Github:

https://github.com/ojh6507/PixelGame

Youtube:

https://www.youtube.com/watch?v=dSSu7duatmk&list=PLWfZ1pyQKoSLPlClHc8V4TUMj-BpvatOo

Android Studio, 로그라이트 (Java)

"GameView코드"

GameView는 커스텀 뷰로, 게임 루프와 게임 상태, 사용자 상호작용을 처리합니다.

Choreographer.FrameCallback을 사용해 매 프레임마다 doFrame 함수를 호출했습니다.
이를 통해 안드로이드 시스템의 프레임 동기화와 정확히 일치하게 되어 효율적으로 처리할 수 있었습니다.

```java
private void scheduleUpdate() { Choreographer.getInstance().postFrameCallback(this); }

JeongHun
@Override
public void doFrame(long nanos) {
    long elapsedNanos = nanos - previousNanos;
    elapsedSeconds = elapsedNanos / 1_000_000_000f;
    if (previousNanos != 0) {
        update();
    }
    invalidate();
    if (running) {
        scheduleUpdate();
    }
    previousNanos = nanos;
};
1 usage  JeongHun
private void update() {
    Scene scene = Scene.top();
    if (scene != null) {
        scene.update(elapsedSeconds);
    }
}
```

Scene의 draw함수를 호출하여 게임 오브젝트들을 화면에 그립니다.

```java
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    canvas.drawColor(Color.BLACK);
    Scene scene = Scene.top();
    if (scene == null) {
        return;
    }
    canvas.save();
    Metrics.concat(canvas);
    if (BuildConfig.DEBUG) {
        //canvas.drawRect(Metrics.borderRect, borderPaint);
    }
    scene.draw(canvas);
    canvas.restore();

    if (BuildConfig.DEBUG) {
        //int fps = (int) (1.0f / elapsedSeconds);
        //int count = scene.count();
        //canvas.drawText("FPS: " + fps + " objs: " + count, 100f, 200f, fpsPaint);
    }
}
```

사용자 터치 이벤트와 게임 상태

```java
@Override
public boolean onTouchEvent(MotionEvent event) {
    Scene scene = Scene.top();
    if (scene != null) {
        boolean handled = scene.onTouch(event);
        if (handled) return true;
    }
    return super.onTouchEvent(event);
}

1 usage  JeongHun
public void onBackPressed() {
    Scene scene = Scene.top();
    if (scene == null) {
        Scene.finishActivity();
        return;
    }
    boolean handled = scene.onBackPressed();
    if (handled) return;

    Scene.pop();
}

1 usage  JeongHun
public void pauseGame() {
    running = false;
    Scene.pauseTop();
}

1 usage  JeongHun
public void resumeGame() {
    if (running) return;
    running = true;
    previousNanos = 0;
    scheduleUpdate();
    Scene.resumeTop();
}

1 usage  JeongHun
public void destroyGame() { Scene.popAll(); }
```

GameView코드 전체:
https://github.com/ojh6507/PixelGame/blob/main/Project/app/src/main/java/kr/ac/tukorea/ge/spg/ojh/framework/view/GameView.java

Android Studio, 로그라이트 (Java)

"Scene코드" Scene.java에서 Scene을 스택으로 관리합니다.

layers 리스트로 Scene내부의 오브젝트를 레이어 단위로 관리합니다.

```
5 usages  JeongHun
protected <E extends Enum<E>> void initLayers(E enumCount) {
    layers = new ArrayList<>();
    int layerCount = enumCount.ordinal();
    for (int i = 0; i < layerCount; i++) {
        layers.add(new ArrayList<>());
    }
}

10 usages  JeongHun
public <E extends Enum<E>> ArrayList<IGameObject> objectsAt(E layerEnum) {
    return layers.get(layerEnum.ordinal());
}

 JeongHun
public <E extends Enum<E>> void add(E layer, IGameObject gameObject) {
    ArrayList<IGameObject> objects = layers.get(layer.ordinal());
    objects.add(gameObject);
}

7 usages  JeongHun
public <E extends Enum<E>> void remove(E layer, IGameObject gameObject) {
    ArrayList<IGameObject> objects = layers.get(layer.ordinal());
    objects.remove(gameObject);
    if (gameObject instanceof IRecyclable) {
        RecycleBin.collect((IRecyclable) gameObject);
    }
}
```

update함수와 draw 함수

```
5 overrides  JeongHun
public void update(float elapsedSeconds) {
    for (ArrayList<IGameObject> objects : layers) {
        int count = objects.size();
        for (int i = count - 1; i >= 0; i--) {
            IGameObject gameObject = objects.get(i);
            gameObject.update(elapsedSeconds);
        }
    }
}

5 usages
protected static Paint bboxPaint;
 JeongHun
public void draw(Canvas canvas) { draw(canvas,  index: stack.size() - 1); }
 JeongHun
protected static void draw(Canvas canvas, int index) {
    Scene scene = stack.get(index);
    if (scene.isTransparent() && index > 0) {
        draw(canvas,  index: index - 1);
    }

    if (scene.clipsRect()) {
        canvas.clipRect( left: 0,  top: 0, Metrics.width, Metrics.height);
    }
    for (ArrayList<IGameObject> objects: scene.layers) {
        for (IGameObject gobj : objects) {
            gobj.draw(canvas);
        }
    }
    if (Scene.drawsDebugInfo) {
        if (bboxPaint == null) {
            bboxPaint = new Paint();
            bboxPaint.setStyle(Paint.Style.STROKE);
            bboxPaint.setColor(Color.RED);
        }
        for (ArrayList<IGameObject> objects: scene.layers) {
            for (IGameObject gobj : objects) {
                if (gobj instanceof IBoxCollidable) {
                    RectF rect = ((IBoxCollidable) gobj).getCollisionRect();
                    canvas.drawRect(rect, bboxPaint);
                }
            }
        }
    }
}
```

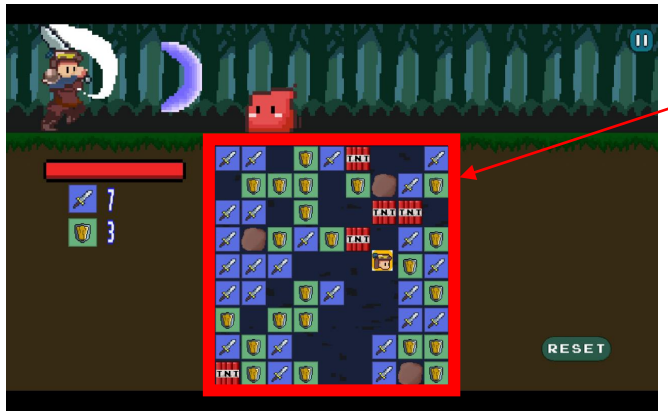
Scene 전환 담당

```
public void push() { push( scene: this); }

1 usage  JeongHun
public static void change(Scene scene) {
    Scene prev = top();
    if (prev != null) {
        scene.onEnd();
    }
    int topIndex = stack.size() - 1;
    stack.set(topIndex, scene);
    scene.onStart();
}

no usages  JeongHun
public void change() { change( scene: this); }

2 usages  JeongHun
public static void pop() {
    Scene scene = top();
    if (scene == null) {
        Log.e(TAG,  msg: "Scene Stack is empty in Scene.pop()");
        return;
    }
    scene.onEnd();
    stack.remove(scene);

    scene = top();
    if (scene == null) {
        Log.i(TAG,  msg: "Last scene is being popped");
        finishActivity();
        return;
    }
    scene.onResume();
}

1 usage  JeongHun
public static void popAll() {
    int count = stack.size();
    for (int i = count - 1; i >= 0; i--) {
        Scene scene = stack.get(i);
        scene.onEnd();
    }
    stack.clear();
    finishActivity();
}
```

Scene코드 전체:
https://github.com/ojh6507/PixelGame/blob/main/Project/app/src/main/java/kr/ac/tukorea/ge/spg/ojh/framework/scene/Scene.java

Android Studio, 로그라이트 (Java)

"Tile Generator 코드"



Tile Generator는 타일 아이템들과 player를 중복 없이 랜덤한 위치에 배치하고 생성합니다.

중복 없이 좌표를 랜덤하게 생성하기 위해 HashSet을 사용했습니다.
Set에 없는 좌표값일 때만 Scene layer에 추가했습니다.

플레이어의 위치를 랜덤으로 배치하고,
각 스테이지 난이도를 위해 스테이지마다 생성될 오브젝트 개수를 설정했습니다.

```java
public void ResetGenerateObjects(WarriorHead warriorHead){

    usedPositions.clear();
    int col = random.nextInt( bound: 9);
    randomX = leftBound + (col * 0.64f);
    int row = random.nextInt( bound: 9);
    randomY = upperBound + (row * 0.66f);
    String posKey = generatePositionKey(randomX, randomY);
    warriorHead.setPosition(randomX,randomY);
    usedPositions.add(posKey);
    int stage = scene.getStage();
    if(stage == 1) {
        OBSTACLE_COUNT = 3;
        BOMB_COUNT = 6;
    }else  if(stage == 2) {
        OBSTACLE_COUNT = 4;
        BOMB_COUNT = 7;
    }
    else  if(stage == 3) {
        OBSTACLE_COUNT = 5;
        BOMB_COUNT = 7;
    }
    generateObjects(OBSTACLE_COUNT, MainScene.Layer.obstacle);
    generateObjects(BOMB_COUNT, MainScene.Layer.bomb);
    generateObjects( count: TOTAL_COUNT -PLAYER_COUNT - OBSTACLE_COUNT - BOMB_COUNT + 1, MainScene.Layer.item);


}
```

```java
private void generateObjects(int count, MainScene.Layer layerType) {

    int attempts = 0; // 현재 시도 횟수
    int attemptLimit = 20;
    int objectsPlaced = 0;
    if (scene == null) return;
    while (objectsPlaced < count && attempts < attemptLimit) {
        for (int i = 0; i < count; i++) {
            calculatePositionX();
            calculatePositionY();
            String posKey = generatePositionKey(randomX, randomY);
            if (!usedPositions.contains(posKey)) {
                scene.add(layerType, createObject(layerType, randomX, randomY));
                 ++objectsPlaced;
            }
        }
        ++attempts;
    }

}
```

```java
1 usage   Oh Jeong Hun
private IGameObject createObject(MainScene.Layer layerType, float x, float y) {
    String posKey = generatePositionKey(x,y);
    if (layerType == MainScene.Layer.obstacle) {
        usedPositions.add(posKey);
        return new Obstacle(x, y);
    } else if (layerType == MainScene.Layer.bomb) {
        usedPositions.add(posKey);
        return new Bomb(x, y);
    }
    else if (layerType == MainScene.Layer.item) {
        if( random.nextInt( bound: 2) == 0){
            usedPositions.add(posKey);
            return new SwordItem(x, y);
        }
        else {
            usedPositions.add(posKey);
            return new ShieldItem(x, y);
        }
    }
    return null;
}
```

```java
3 usages   Oh Jeong Hun
private String generatePositionKey(float x, float y) {
    return String.format("%.2f,%.2f", x, y);
}
```

Hashkey는 String으로

Tile Generator 코드 전체:
https://github.com/ojh6507/PixelGame/blob/main/Project/app/src/main/java/kr/ac/tukorea/ge/spg/ojh/pixelgame/game/TileGenerator.java

Android Studio, 로그라이트 (Java)

“충돌체크”

게임에서 회전이 발생하지 않아 AABB 충돌을 사용했습니다.

```
public class CollisionHelper {
    4 usages  JeongHun
    public static boolean collides(IBoxCollidable obj1, IBoxCollidable obj2) {
        RectF r1 = obj1.getCollisionRect();
        RectF r2 = obj2.getCollisionRect();

        if (r1.left > r2.right) return false;
        if (r1.top > r2.bottom) return false;
        if (r1.right < r2.left) return false;
        if (r1.bottom < r2.top) return false;

        return true;
    }
}
```

CollisionChecker의 update함수에서 매 tick마다 충돌 체크합니다.

```
public void update(float elapsedSeconds) {
    ArrayList<IGameObject> obstacles = scene.objectsAt(MainScene.Layer.obstacle);
    for(int o = obstacles.size() - 1; o>=0; o--){
        Obstacle obs = (Obstacle) obstacles.get(o);
        if(CollisionHelper.collides(obs, this.warriorHead)){
            RectF obstacleRect = obs.getCollisionRect();
            RectF headRect = this.warriorHead.getCollisionRect();

            updateDirectionAfterCollision(obstacleRect, headRect);
            break;
        }

    }
    ArrayList<IGameObject> bombs = scene.objectsAt(MainScene.Layer.bomb);
    for(int b = bombs.size() - 1; b>=0; b--){
        Bomb bomb = (Bomb) bombs.get(b);
        if(CollisionHelper.collides(bomb, this.warriorHead)){
            this.warriorHead.Stop();
            bomb.ExplosionEffect();
            break;
        }

    }
    ArrayList<IGameObject> items = scene.objectsAt(MainScene.Layer.item);
    for (int i = items.size() - 1; i >= 0; i--) {
        IGameObject gobj = items.get(i);
        if (!(gobj instanceof Item)) {
            continue;
        }
        Item item = (Item) gobj;
        if (CollisionHelper.collides(warriorHead, item)) {
            Sound.playEffect(R.raw.item);
            if(item instanceof SwordItem)
                warriorHead.PowerUp(1);
            else
                warriorHead.DefUp( df: 1);
            scene.SetAtkScore(warriorHead.GetEarnPower());
            scene.SetDefScore(warriorHead.GetEarnDef());
            scene.remove(MainScene.Layer.item, gobj);
        }
    }

    ArrayList<IGameObject> enemies = scene.objectsAt(MainScene.Layer.enemy);
    for (int e = enemies.size() - 1; e >= 0; e--) {
        Slime enemy = (Slime)enemies.get(e);

        if(enemy.dead) {
            continue;
        }
```

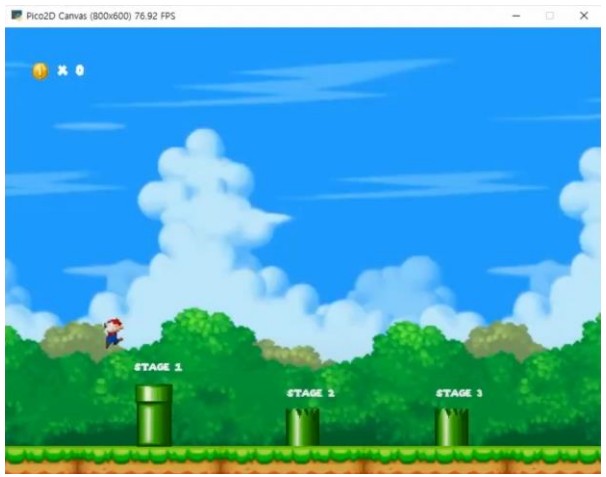
충돌 체크될 객체들은
IBoxCollidable 인터페이스로 확장했습니다.

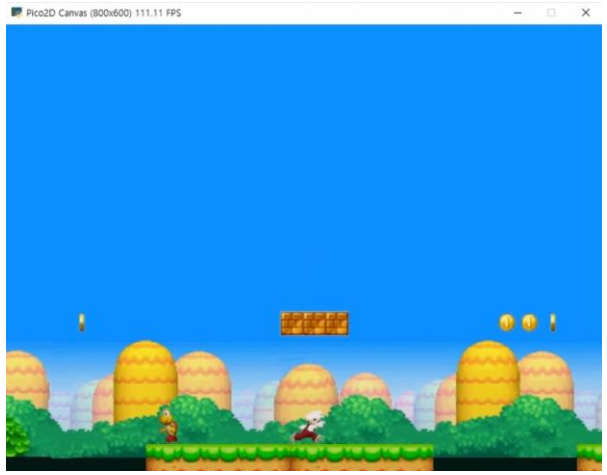
```
18 usages  9 implementations  JeongHun
public interface IBoxCollidable {
    7 implementations  JeongHun
    public RectF getCollisionRect();
}
```

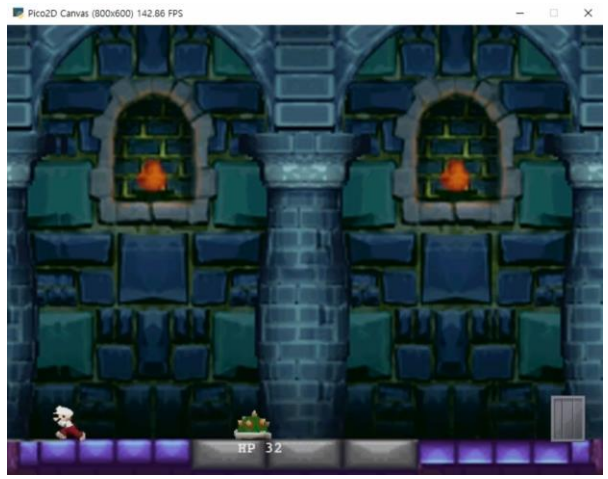
CollisionChecker코드 전체:
https://github.com/ojh6507/PixelGame/blob/main/Project/app/src/main/java/kr/ac/tukorea/ge/spg/ojh/pixelgame/game/CollisionChecker.java

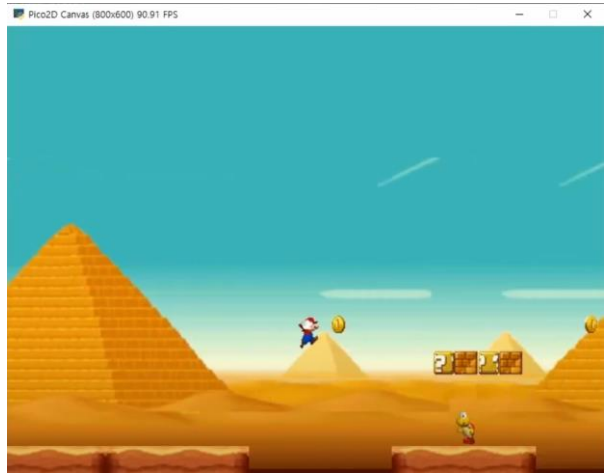
Pico2d, Super Mario (Python)

"Super Mario" 2022/10 ~ 2022/12, 1인 개발, 클라이언트 프로그래머로 참여









스테이지를 하나씩 클리어하며 진행하는 2D 플랫폼 게임

2D 게임 프로그래밍 텀 프로젝트

Pico2d, Super Mario (Python)

“Super Mario”

주요 작업 (본인 파트)

- 모든 클라이언트 코드 제작

Github:

https://github.com/ojh6507/2019180024_2DGP_TermProject/tree/main/SuperMario

Youtube:

https://www.youtube.com/watch?v=rdnpZRyUbSU&list=PLWfZ1pyQKoSLnoRooyCb8KBZsQxHnAP2P

Pico2d, Super Mario (Python)

“game framework” 코드

스택으로 게임 씬 관리했습니다.

인덱스 -1로 액세스하여 항상 마지막 요소에 접근하도록 했습니다

Change_state함수 호출로 scene전환이 이뤄집니다.

```python
import time
frame_time = 0.0
class GameState:
    def __init__(self, state):
        self.enter = state.enter
        self.exit = state.exit
        self.pause = state.pause
        self.resume = state.resume
        self.handle_events = state.handle_events
        self.update = state.update
        self.draw = state.draw


running = None
stack = None


def change_state(state):
    global stack
    if (len(stack) > 0):
        # execute the current state's exit function
        stack[-1].exit()
        # remove the current state
        stack.pop()
    stack.append(state)
    state.enter()



def push_state(state):
    global stack
    if (len(stack) > 0):
        stack[-1].pause()
    stack.append(state)
    state.enter()



def pop_state():
    global stack
    if (len(stack) > 0):
        # execute the current state's exit function
        stack[-1].exit()
        # remove the current state
        stack.pop()

    # execute resume function of the previous state
    if (len(stack) > 0):
        stack[-1].resume()



def quit():
    global running
    running = False
```

run함수:

게임 루프를 실행하여 현재 씬의 이벤트 처리, 업데이트, 그리기를 반복합니다.

종료 시 모든 씬들 정리

```python
def run(start_state):
    global running, stack
    running = True
    stack = [start_state]
    start_state.enter()

    current_time = time.time()
    while (running):
        stack[-1].handle_events()
        stack[-1].update()
        stack[-1].draw()
        global frame_time
        frame_time = time.time() - current_time
        frame_rate = 1.0/frame_time
        current_time += frame_time
        # print(f'Frame Time: {frame_time}, Frame Rate: {frame_rate}')

    # repeatedly delete the top of the stack
    while (len(stack) > 0):
        stack[-1].exit()
        stack.pop()
```

game_framework코드 전체:
https://github.com/ojh6507/2019180024_2DGP_TermProject/blob/main/SuperMario/game_framework.py

Pico2d, Super Mario (Python)

"Super Mario" 코드

Stage 맵 관련 코드 입니다.

맵 정보를 INFO 이차원 배열에 맵 객체 타입별로 입력했습니다.

```python
INFO = np.zeros((13, 40))
for i in range(40):
    INFO[0, i] = 1

INFO[0,8] = -1
INFO[0,10] = -1
INFO[0,25] = -1
INFO[0,28] = -1
INFO[0,31] = -1


INFO[1,13] = 1
INFO[0,13] = 2

INFO[1,19] = 1
INFO[0,19] = 2

INFO[2,20] = 1
INFO[1,20] = 2
INFO[0,20] = 2

INFO[1,21] = 1
INFO[0,21] = 2


class Empty_Tile:
    image = None
    def __init__(self, col, row):
        self.x, self.y = row * 192, col * 77
    def get_name(self):
        return 'background'
    def edit_x(self,x):
        self.x-=x
    def get_bb(self):
        return self.x - 98, self.y - 40, self.x + 98, self.y + 200
    def __init__(self,col, row):
        self.x, self.y = row * 192, col * 77


    def update(self):
        pass
    def draw(self):
        pass

    def handle_collision(self, other, group, pos):
        pass


class Floor_Tile(Empty_Tile):
    image = None
    def __init__(self,col, row):
        if Floor_Tile.image == None:
            Floor_Tile.image = load_image('./background/ground.png')
        self.x, self.y = row * 192, col * 78

    def get_bb(self):
        return self.x - 100, self.y - 40, self.x + 100, self.y + 40
    def update(self):
        pass
    def draw(self):
        self.image.draw(self.x, self.y)

    def handle_collision(self,other,group,pos):
        pass


class under_Tile(Floor_Tile):
    image = None

    def __init__(self, col, row):
        if under_Tile.image == None:
            under_Tile.image = load_image('./background/underground.png')
        self.x, self.y = row * 192, col * 78
```

맵 정보 바탕으로 맵 오브젝트들을 생성합니다.

```python
def set_world():
    for col in range(len(world_menu.INFO)):
        for row in range(len(world_menu.INFO[col])):

            if world_menu.INFO[col][row] == -1:
                server.empty.append(world_menu.Empty_Tile(col, row))

            elif world_menu.INFO[col][row] == 1:
                server.ground.append(world_menu.Floor_Tile(col, row))

            elif world_menu.INFO[col][row] == 2:
                server.ground.append(world_menu.under_Tile(col, row))


def enter():
    server.world = round1.BACKGROUND()
    set_world()
    server.player = character.mario()
    pipe = block.Pipe()
    pipe.activate = True

    server.coin = [block.COIN() for n in range(0, 20)]
    server.itemBox = [block.item_block() for n in range(10)]
    server.bricks = [block.Bricks() for n in range(40)]
    server.goomba = [Goomba.GOOMBA() for i in range(5)]
    server.green = [Koopa.GreenKoopa() for i in range(4)]
    server.red = [Koopa.RedKoopa() for i in range(4)]

    setPos()

    game_world.add_object(server.world, 0)
    game_world.add_object(server.player, 1)
    game_world.add_objects(server.goomba, 1)
    game_world.add_objects(server.green, 1)
    game_world.add_object(pipe, 3)

    game_world.add_objects(server.red, 1)
    game_world.add_objects(server.coin, 2)
    game_world.add_objects(server.itemBox, 2)
    game_world.add_objects(server.bricks, 1)
    game_world.add_objects(server.ground, 3)
    game_world.add_objects(server.empty, 3)

    game_world.add_collision_group(server.player, server.coin, 'player:coin')
    game_world.add_collision_group(server.player, server.itemBox, 'player:item_bl
    game_world.add_collision_group(server.player, server.bricks, 'player:bricks')
```

이 코드를 수정한다면, tiled 프로그램을 사용하여 파일 입출력으로
맵 생성되도록 수정할 것 같습니다

play_state코드 전체:
https://github.com/ojh6507/2019180024_2DGP_TermProject/blob/main/SuperMario/play_state.py

Pico2d, Super Mario (Python)

“Super Mario” 충돌체크 관련 코드

씬에서 콜리전 체크를 합니다.

슈퍼마리오는 아이템 블록들이 존재하기 때문에 상/하/좌/우를 구분하도록 했습니다.

```python
def collide(a,b):
    str = ' '

    la, ba, ra, ta = a.get_bb()
    lb, bb, rb, tb = b.get_bb()
    if la > rb: return False
    if ra < lb: return False
    if ta < bb: return False
    if ba > tb: return False

    if ra >= lb and la <= lb:
        str = 'right'
    elif rb >= la and rb <= ra:
        str = 'left'

    if ((ra - lb >= 1 and lb - la <= 50) or (rb - la >= 1 and ra - rb <= 50) or (ra <= rb and lb <= la)) and (tb >= ba and ta > tb):
        str = 'bottom'
    elif ((ra - lb >= 1 and lb - la <= 15) or (rb - la <= 15 and ra - rb <= 15) or (ra <= rb and lb <= la)) and (ta - bb < 20 and bb > ba):
        str = 'top'


    return True, str
```

슈퍼마리오 캐릭터 콜리전 처리 함수

```python
def handle_collision(self, other, group, pos):
    if not self.die:
        if group == 'player:coin':
            server.coin_count += 1

        elif group == 'player:item_block':
            if pos == 'bottom':
                la, ba, ra, ta = self.get_bb()
                lb, bb, rb, tb = other.get_bb()
                if tb > ba:
                    self.y += 2

                self.Onground = True
                if abs(self.x - other.x) <= 15:
                    self.Onground = True
                    self.jump = False
                    self.y -= self.pre_velocity * JUMP_SPEED_PPS * game_framework.frame_time
                    self.Y_velocity = 0
                    self.pre_velocity = 0

            if pos == 'right':
                self.x -= 15
                self.x_dir = 0

            if pos == 'left':
                self.x += 15
                self.x_dir = 0

            if pos == 'top':
                self.Onground = False
                if self.Y_velocity > 0:
                    self.Y_velocity *= -1
                    self.y += self.Y_velocity * JUMP_SPEED_PPS * game_framework.frame_time

        elif group == 'player:bricks' and (other.available or other.op=='solid'):
            if pos == 'bottom':
                la, ba, ra, ta = self.get_bb()
                lb, bb, rb, tb = other.get_bb()
                if tb > ba:
                    self.y += 2

                self.jump = False
                self.Y_velocity = 0
                self.pre_velocity = 0
                self.y -= self.pre_velocity * JUMP_SPEED_PPS * game_framework.frame_time


            if pos == 'right':
                self.x_dir = 0
                self.x -= 10

            if pos == 'left':
                self.x_dir = 0
                self.x += 10

            if pos == 'top':
                if self.Y_velocity > 0:
                    self.Y_velocity *= -1
                    self.y += self.Y_velocity * JUMP_SPEED_PPS * game_framework.frame_time


        elif group == 'player:mushroom':
            self.powerup.play()
            self.y += 10
            self.mario_size = 'Normal'
            self.jump_height = 13


        elif group == 'player:flower':
            self.powerup.play()
            self.mario_size = 'Normal'
            self.flower = True
            self.jump_height = 13

        elif group == 'player:ground':
            if pos == 'bottom':
                la, ba, ra, ta = self.get_bb()
                lb, bb, rb, tb = other.get_bb()
                if tb > ba:
                    self.y+= 2
                self.jump = False
                self.Y_velocity = 0
                self.pre_velocity = 0
                self.y -= self.pre_velocity * JUMP_SPEED_PPS * game_framework.frame_time
```

그룹 단위로 충돌 처리합니다

예를 들어 그룹이 player:mushroom이라면 size를 small에서 Normal로 바꿔서 슈퍼마리오 캐릭터의 sprite를 변경하도록 했습니다.

이 코드를 개선한다면, 함수 분리와 코드 구조화를 통해 중복을 줄이고,

각 객체에 맞는 충돌 처리를 효율적으로 관리할 것 같습니다.

씬 enter함수에서 게임 월드에 플레이어, 적, 아이템, 오브젝트 등을 추가하고,

각 객체 간의 충돌 그룹을 정의합니다.

```python
def enter():
    server.world = round1.BACKGROUND()
    set_world()
    server.player = character.mario()
    pipe = block.Pipe()
    pipe.activate = True

    server.coin = [block.COIN() for n in range(0, 20)]
    server.itemBox = [block.item_block() for n in range(10)]
    server.bricks = [block.Bricks() for n in range(40)]
    server.goomba = [Goomba.GOOMBA() for i in range(5)]
    server.green = [Koopa.GreenKoopa() for i in range(4)]
    server.red = [Koopa.RedKoopa() for i in range(4)]

    setPos()

    game_world.add_object(server.world, 0)
    game_world.add_object(server.player, 1)
    game_world.add_objects(server.goomba, 1)
    game_world.add_objects(server.green, 1)
    game_world.add_object(pipe, 3)

    game_world.add_objects(server.red, 1)
    game_world.add_objects(server.coin, 2)
    game_world.add_objects(server.itemBox, 2)
    game_world.add_objects(server.bricks, 1)
    game_world.add_objects(server.ground, 3)
    game_world.add_objects(server.empty, 3)

    game_world.add_collision_group(server.player, server.coin, 'player:coin')
    game_world.add_collision_group(server.player, server.itemBox, 'player:item_block')
    game_world.add_collision_group(server.player, server.bricks, 'player:bricks')
    game_world.add_collision_group(server.player, server.ground, 'player:ground')
    game_world.add_collision_group(server.player, server.goomba, 'player:goomba')
    game_world.add_collision_group(server.player, server.red, 'player:red')
    game_world.add_collision_group(server.player, server.green, 'player:green')
    game_world.add_collision_group(server.goomba, server.ground, 'goomba:ground')
    game_world.add_collision_group(server.goomba, server.itemBox, 'goomba:itemBox')
    game_world.add_collision_group(server.goomba, server.bricks, 'goomba:bricks')
    game_world.add_collision_group(server.green, server.ground, 'green:ground')
    game_world.add_collision_group(server.red, server.ground, 'red:ground')
    game_world.add_collision_group(server.red, server.empty, 'red:empty')
    game_world.add_collision_group(server.player, pipe, 'player:pipe')

    game_world.add_collision_group(None, server.ground, 'fire:ground')
    game_world.add_collision_group(server.goomba, None,'fire:goomba')
    game_world.add_collision_group(server.red, None, 'fire:red')
    game_world.add_collision_group(server.green, None, 'fire:green')
    game_world.add_collision_group(server.itemBox, None, 'fire:itembox')
    game_world.add_collision_group(server.bricks, None, 'fire:bricks')
    game_world.add_collision_group(pipe, None, 'fire:pipe')

    # game_world.add_collision_group(server.player, server.stair, 'player:stair')
```

Pico2d, Super Mario (Python)

“character” 코드

슈퍼마리오 캐릭터는 상태기반 캐릭터 입니다.

현재 상태의 do 함수 호출

```
def update(self):
    self.ACTION_PER_TIME = 1.0 / self.TIME_PER_ACTION
    self.cur_state.do(self)

    self.delay += game_framework.frame_time
    self.delay_draw += 1

    if self.y < 0:
        self.event_que.clear()

    if self.event_que:
        event = self.event_que.pop()

        if event == SPACE:
            if not self.jump:
                self.frame = 0
                self.jump = True
                self.Y_velocity = self.jump_height
                self.jump_music.play()
        elif event == SHIFTD:
            self.frame = 0
            self.Run = True
        elif event == DD:
            self.godown = True
        elif event == UD:
            self.door_open = True
        if event == SHIFTU:
            self.Run = False

        self.cur_state.exit(self,event)
        try:
            self.cur_state = next_state[self.cur_state][event]
        except KeyError:
            print('Error: ', self.cur_state.__name__,' ', event_name[event])
        self.cur_state.enter(self, event)

    self.x = clamp(0,self.x,800)
    if self.tempx != 0:
        self.x = self.tempx
    self.jump_func()

    if self.invincibility:
        self.timer -= game_framework.frame_time
        if self.timer <= 0:
            self.invincibility = False
            self.delay = 1.1
            self.timer = 1.0

    if self.die:
        self.delay = 1.1
        self.jump = True
        self.Run = False
        self.cur_state = DIE

    if self.clear:
        self.cur_state = Clear_movement
        self.jump = True
        self.Y_velocity = self.Y_gravity
        self.clear_voice.play()
```

event_que에 저장된 키 이벤트 처리하여
현재 상태 전환 및 동작 수행

IDLE 상태

```
class IDLE:
    def enter(self, event):
        self.x_dir = 0
        self.frame = 0
        self.Run = False
        self.TIME_PER_ACTION = 1
        if self.mario_size == 'Small':

            self.perframe = 30
            self.action = 0
            self.height = 35

        elif self.mario_size == 'Normal':

            self.perframe = 50
            self.action = 0
            self.height = 65

    def exit(self, event):
        if event == ATTACK:
            self.Fire_Ball()

    def do(self):
        if not self.jump:
            if self.mario_size == 'Small':
                self.clip = 76

            elif self.mario_size == 'Normal':
                self.clip = 79
        else:

            if self.mario_size == 'Small':
                self.clip = 30
                self.TIME_PER_ACTION = 1
            elif self.mario_size == 'Normal':
                self.clip = 18
                self.TIME_PER_ACTION = 0.5

        if self.face_dir == 1:
            self.reflect = ' '
        else:
```

Walk 상태

```
class WALK:
    def enter(self, event):
        self.frame = 0
        self.walk = True
        if event == RD:
            self.face_dir = 1
            self.  (variable) face_dir: Literal[-1]
        elif even
            self.face_dir = -1
            self.x_dir = self.face_dir
        elif event == RU:
            self.face_dir = -1
            self.x_dir = self.face_dir
        elif event == LD:
            self.face_dir = 1
            self.x_dir = self.face_dir
        self.TIME_PER_ACTION = 1

    def exit(self,event):
        self.walk = False
        if event == ATTACK:
            self.Fire_Ball()
        self.velocity = 1

    def do(self):
        if self.Run:
            self.TIME_PER_ACTION = 0.4
            self.velocity = 3
            if self.mario_size == 'Small':
                # self.image = load_image('smario_run.png')
                self.clip = 13

            if self.mario_size == 'Normal':
                # if not self.flower:
                #     self.image = load_image('run_fast.png')
                # else:
                #     self.image = load_image('flower_run_fast.png')
                self.clip = 18

        if not self.Run:
            self.TIME_PER_ACTION = 1
            self.velocity = 1
            if self.mario_size == 'Small':
                self.clip = 27
            if self.mario_size == 'Normal':
                self.clip = 25

        if self.jump:
```

DIE 상태

```
class DIE:
    def enter(self, event):
        self.die = True

    def exit(self, event):
        pass

    def do(self):

        self.Onground = False
        self.image = load_image('player/gameover_mario.png')
        self.clip = 13
        self.height = 60
        self.perframe = 50
        self.frame = (self.frame + self.ACTION_PER_TIME * self.clip * game_framework.frame_time) % self.clip


    def draw(self):
        if self.mario_size == 'Small':
            self.image.clip_composite_draw(int(self.frame) * self.perframe, 0, self.perframe, self.height, 0,
                                           self.reflect, self.x, self.y, 40, 40)

        elif self.mario_size == 'Normal':
            self.image.clip_composite_draw(int(self.frame) * self.perframe, 0, self.perframe, self.height, 0,
                                           self.reflect, self.x, self.y, 50, 50)
```

Clear 상태

```
class Clear_movement:
    def enter(self,event):
        self.Y_velocity = self.jump_height

        pass
    def exit(self, event):
        pass
    def do(self):
        self.Onground = False
        self.image = load_image('player/clear_mario.png')
        self.clip = 9
        self.height = 37
        self.perframe = 33
        self.y += 5 * JUMP_SPEED_PPS * game_framework.frame_time * 2
        self.frame = (self.frame + self.ACTION_PER_TIME * self.clip * game_framework.frame_time) % self.clip

    def draw(self):
        if self.mario_size == 'Small':
            self.image.clip_composite_draw(int(self.frame) * self.perframe, 0, self.perframe, self.height, 0,
                                           self.reflect, self.x, self.y, 50, 40)

        elif self.mario_size == 'Normal':
            self.image.clip_composite_draw(int(self.frame) * self.perframe, 0, self.perframe, self.height, 0,
                                           self.reflect, self.x, self.y, 50, 50)
```

키 입력과 상태 전이 (딕셔너리 사용)

```
next_state = {
    IDLE: {RU: WALK, LU: WALK, RD: WALK, LD: WALK, ATTACK: IDLE, SHIFTD: IDLE, SHIFTU: IDLE, SPACE: IDLE, DD:IDLE, DU:IDLE, UD:IDLE},
    WALK: {RU: IDLE, LU: IDLE, RD: IDLE, LD: IDLE, ATTACK: WALK, SHIFTD: WALK, SHIFTU: WALK, SPACE: WALK, DD:WALK, DU:WALK, UD:WALK},
    DIE: {RU: DIE, LU: DIE, RD: DIE, LD: DIE, ATTACK: DIE, SHIFTD: DIE, SHIFTU: DIE, SPACE: DIE, DD:DIE, DU:DIE, UD:DIE},
    Clear_movement: {RU: Clear_movement, LU: Clear_movement, RD: Clear_movement, LD: Clear_movement, ATTACK: Clear_movement, SHIFTD: Clear_movement,
                     SHIFTU: Clear_movement, SPACE: Clear_movement, DD:Clear_movement, DU:Clear_movement, UD:Clear_movement}

}
```

character.py코드 전체:
https://github.com/ojh6507/2019180024_2DGP_TermProject/blob/main/SuperMario/player/character.py