DPhysic 사용 설명서

프로젝트 세팅

솔루션 탐색기에서 솔루션 오른 클릭 후 기존 프로젝트 추가에서 DPhysics.vcxproj 추가

프로젝트 라이브러리에

DLL 파일을 사용하므로 속성 - 디버깅 - 환경에

PATH=$(ProjectDir)\Library\$(Configuration);%PATH%; 입력 (DLL 파일 사용 안 하면 안해도 됨)

C/C++ - 일반 추가 포함 디렉터리에

($(SolutionDir) 또는 $(ProjectDir) 위치 기준 Dphysics 있는 상대위치)\Dphysics

(상대위치)\Dphysics\Library\physx

(상대위치)\ Library\recast 추가

링커 – 일반 추가 포함 라이브러리에

(상대위치)\DPhysics\Library\$(Configuration) 추가

메인 루프 Cpp 파일 상단에 다음과 같이 매크로 입력과 네임스페이스 선언을 한다.

//PhysX 헤더

#include "PhysX.h"

//물리엔진 헤더

#pragma comment(lib,"DPhysics.lib")

#include "PhysicsEngine.h"

//물리 엔진 네임스페이스

using namespace PhysicsEngine;

PhysicsEngine

물리 엔진 네임스페이스

PhysicsEngine::PhysicsEnginMain은 싱글턴이므로

GetInstance()를 사용해 함수에 접근한다.

void Initialize();

물리 엔진 초기화 함수

void Update();

물리 엔진 시뮬레이션 업데이트 함수 초당 60번씩 업데이트 한다

후술할 Component 별 FixedUpdate 실행 후, Update 실행 전에 시뮬레이션을 돌려야 정상 작동한다.

void Finalize();

물리 엔진 종료 함수

PhysicsEngineInterface

사용할 Object별 헤더파일 또는 소스파일 상단에

#include “PhysicsEningeInterface.h” 추가

RRIGIDACTOR\_PTR 물리엔진 내부 강체 오브젝트의 포인터, 함수에 입력함으로써 어떤 내부 오브젝트 인지 특정하고 정보를 변경하거나 가져올 수 있다.

CCT\_PTR 캐릭터 컨트롤러(CCT)의 포인터, 직접 유저가 정의한 움직임을 사용할 때 쓴다. 마찬가지로 함수에 입력함으로써 어떤 내부 오브젝트 인지 특정하고 정보를 변경하거나 가져올 수 있다.

CreateRigidStaticXXX

Static Actor를 생성, 물리 시물레이션에 의해 움직이지 않고 밀어내기만 하는 강체 Actor를 생성

박스, 구, 캡슐, 사각형, 바닥, 삼각형 매시 모양을 생성할 수 있다.

CreateStaticActorFromMeshData(pos, rot, mesh);

rcMeshLoaderObj 로 mesh 데이터를 읽은 다음 PhysicsEngineInterface::TRIANGLEMESH 에 버텍스/페이스 개수와 버텍스/페이스(인덱스) 데이터를 넣어서 사용 StaticMesh에 사용예시 있음

CreateRigidDynamicXXX

Dynamic Actor를 생성, 물리 시물레이션에 의해 밀어내고 움직이는 하는 강체 Actor를 생성

박스, 구, 캡슐 모양을 생성할 수 있다.

CreateRigidKinematicXXX

Kinematic Actor를 생성(실제로는 Dynamic Actor와 동일하지만, 일부 플래그(옵션)이 다르다), 물리 시물레이션에 영향 받지 않는 강체 Actor를 생성, 질량(mass)이 없다. 사용자가 원하는 인위적인 움직임을 구현할 때 사용한다.

박스, 구, 캡슐 모양을 생성할 수 있다.

UpdateRigidDynamic

Dynamic Actor의 Transform(위치, 회전)을 업데이트 할 때 사용한다. 물리 시뮬레이션에 의한 작용을 보려면 PhysicsEngineMain::Update() 이전에 사용하는 것이 좋다.

UpdateRigidKinematic

Kinematic Actor의 Transform(위치, 회전)을 업데이트 할 때 사용한다. 물리 시뮬레이션에 의한 작용을 보려면 PhysicsEngineMain::Update() 이전에 사용하는 것이 좋다.

GetTransformRigidActor

모든 RigidActor의 Transform 정보를 가져온다.

물리 시뮬레이션에 의한 결과를 보려면 PhysicsEngineMain::Update() 이후에 사용하는 것이 좋다.

DeleteRigidActor

RigidActor를 삭제

CreateCCTBox

CreateCCTCapsule

Character Controller(CCT)를 생성, 박스 또는 캡슐의 형태로 생성이 가능하다. 실제 렌더링 되는 물체에 비례하여 크기를 설정하는 것이 좋다.

UpdatePositionCCT

Character Controller의 위치를 변경하는 함수, 위치가 아닌 현재 위치에서 얼마나 이동 할 것 인지를 미리 정한 변위 displacement와 이동하는 시간 dTime 을 받아 다른 강체 또는 충돌체와 충돌하는지 여부 등을 질의(Query)하고 결과에 따라 밀어내기, 충돌 등의 정의된 행동을 실행한다.

CCT에서 회전은 계산하지 않는다. 따라서 렌더링 시 회전된 결과를 보고 싶다면 따로 회전을 시켜줘야 한다.

PhysicsEngineMain::Update() 이전에 사용하는 것이 좋다.

GetPositionCCT

Character Controller 위치를 갖고 온다.

PhysicsEngineMain::Update() 이후에 사용하는 것이 좋다.

DeleteCCT

Character Controller 제거

기타 기능

ConsoleWindow 디버그용 콘솔 기능

MeshLoaderObj 파일 경로 문자열을 입력받아 obj 모델 파일을 로딩한다.

사용 예)

rcMeshLoaderObj\* pMeshLoader = new rcMeshLoaderObj();

pMeshLoader->load("Media/Model/map\_re\_re\_re.obj");

DPhysics\ExampleGame\Media\Model에 테스트용 obj 파일 첨부 되어있음

원하는 경로에 복사 후 사용

PVD

물리 actor를 시각화 해주는 디버거

DPhysics\PVD에 첨부된 설치파일 실행해서 설치

디버그 빌드 실행하기전에 설치된 PVD 실행해서 사용하면 실행 중인 프로세스랑 tcp/ip 로 통신 해서 물리 actor를 실시간으로 보여준다.

DPhysics 업데이트 정리, 추가기능 설명 2021. 03. 23

Create 함수들은 이제 PhysicsEngine::ISimulationEventCallback\* pOwnerCallback을 추가로 입력받는다. 각 Actor, CCT를 사용하는 유저 오브젝트에서 로직 단계에서 정의한 충돌 이벤트 함수에 접근하기 위해서다.

충돌 이벤트 처리

충돌 필터 설정 함수

SetupCollisionFilter(const RRIGIDACTOR\_PTR dest, FILTERGROUP::Enum filterGroup, FILTERGROUP::Enum filterMask)

SetupCollisionFilterCCT( const CCT\_PTR dest, FILTERGROUP::Enum filterGroup, FILTERGROUP::Enum filterMask)

게임 로직 상에서 원하는 충돌 패턴에 한해서 충돌 이벤트 함수(ISimulationEventCallback의 함수)를 사용하려면 충돌 필터가 사전에 설정이 되어야한다.

먼저, PhysicsEngineInterface 네임 스페이스 상단의 FILTERGROUP에 원하는 그룹을 비트 << 연산을 통해 정의한다.

함수 호출 시 대상 객체(dest)에 자신이 속하는 충돌 그룹과(filterGroup)과 자신과 충돌 시 충돌 이벤트 처리를 하고 싶은 다른 그룹(filterMask)을 설정한다. 내부에서는 설정된 filterGroup 과 filterMask 간에 비트 & 연산을 통해 설정한 충돌 패턴를 검출한다. 충돌 이벤트 함수 호출은 충돌이 일어난 객체 쌍(pair)단위로 이루어진다.

SetTriggerShape(RRIGIDACTOR\_PTR&, bool)

생성시 기본적으로 내부 shape의 플래그(eTRIGGER)가 false인 상태이므로 OnTrigger 함수를 사용하려면 true로 set해주면 된다.

SetSimulationShape(RRIGIDACTOR\_PTR&, bool)

반대로 시뮬레이션에 의해 움직이고, OnCollision 이벤트 함수를 사용하려면 true로 set 해주면 된다.

ISimulationEventCallback을 상속한 클래스에서 OnXXXXXX 함수를 오버라이드 하여 사용

구현에서는 ISimulationEventCallback\*을 상속받은 CollsionObject\*로 down casting 하여 CollsionObject\*를 입력받는 함수를 다시 호출하게 사용하였음

사용 예시

//ExampleGame\CollsionObject.cpp

void CollisionObject::OnCollisionEnter(ISimulationEventCallback\* ohter)

{

OnCollisionEnter(dynamic\_cast<CollisionObject\*>(ohter));

}

ISimulationEventCallback\* other는 충돌한 상대 객체 포인터

//ExampleGame\ HeroObject.cpp

void HeroObject::OnCollisionEnter(CollisionObject\* ohter)

{

//여기서 충돌시 실행할 로직을 입력

}

void CollisionObject::OnControllerHit(PhysicsEngine::CharaterControllerHit& hit)

{

OnControllerHitCollision(hit); //전달할 매개변수가 같으므로 하위클래스에서 실행할 가상 //함수의 이름을 다르게 변경함

}

CCT - CCT 충돌의 경우 CharaterControllerHit로 충돌 정보 자체를 입력 받아 사용

충돌 한 자신은 controller, 상대는 otherController 멤버로 접근

충돌이 감지되면 매 프레임(약 1초 당 60번씩)마다 호출한다. 다른 함수처럼 충돌 상태별로 함수가 구분되어 있지 않아 구분하여 처리하고 싶으면 추가로 객체에 상태를 저장하여 사용하여야 한다. 또한 충돌 검출이 객체 쌍 단위로 이루어 지지 않으므로 상대의 충돌처리에 약간의 시간이 걸린다.

UpdatePosition 함수 충돌 방향 리턴기능 추가,

CCT\_COLLISION\_DIR::Enum UpdatePositionCCT(const CCT\_PTR dest, Vector3 velocity, float dTime);

충돌 시에는

SIDE(world up 벡터(y+)에 대해 수직인 방향(xz 평면에 대해 평행한 방향), 옆 방향)

UP(world up vector(y+)방향)

DOWN(world down vector(y-)방향)

을 리턴하며

충돌이 없을 때는 NONE이 리턴된다.

GetFootPositionCCT(const CCT\_PTR, Vector3)

CCT의 밑부분을 리턴하는 함수

IsActive(bool), IsActiveCCT()

모든 충돌 연산에 대한 내부 플래그를 전부 활성화/비활성화, 안전한 삭제가 아직 안 구현 되어있어 대신 사용 중

SetSize 함수

RRIGIDACTOR의 크기를 변경(내부 shape의 모양 별 멤버 변경), TriangleMesh는 정해진 모양이 아니므로 scale(배율) 값을 변경

- 충돌 처리 추가 설명 (같이 첨부한 “물리 충돌 처리.pptx” 순서도와 구조도 참고 바람)

CCT와 RigidBody의 구조, 플래그

CharacterController(CCT)는 RigidBody(Kinematic)를 갖고 있음

RigidBody는 자신의 충돌 영역, 모델을 표현하는 Shape를 갖고 있음(Box, Sphere, Capsule, TriangleMesh, Plane…)

Shape는 시뮬레이션 처리에 대한 여부를 저장하는 플래그를 갖고 있음,

영향을 받으려면 Simulation 플래그 on, Trigger 플래그가 Off(Default),

영향을 받지 않으려면 Simulation 플래그 off, Trigger 플래그가 On

SceneQueryShape 플래그는 Scene 쿼리 처리에 대한 여부를 저장, CCT 충돌처리와 관계 있음(Default On)

충돌 패턴에 따른 콜백함수 실행 요약

기본적으로

Trigger - Trigger shape 간 충돌은 안됨

한쪽이 Trigger면 onTrigger가 실행

둘다 Trigger가 아니면(eSMULTATE\_SHAPE) onContact가 실행

예외적으로

RigidStatic - RigidStatic 간 충돌

RigidStatic - Kinematic Actor 충돌 (둘다 트리거인 경우)

하나가 Trigger가 아닌 Kinematic - Kinematic 충돌

//Two jointed rigid bodiesand the joint was defined to disable collision

//Two articulation links if connected through an articulation joint

이면 충돌 이벤트가 발생하지 않음

충돌 콜백함수 호출 시점

physx scene의 업데이트는 크게 두단계로 나누어 진다.

scene과 actor들의 정보를 바탕으로 시간간격을 입력 받아 물리 시뮬레이션을 하는 simulate과

시뮬레이션 된 결과를 바탕으로 충돌처리나 미리 설정한 콜백함수를 실행하는 fetchResult가 있다.

RigidActor 간 충돌 결과에 따른 콜백함수가 호출되는 시점은 fetchResult에서 호출된다. 하지만 요청은 simulate에서 한다.

CCT의 경우 Scene 쿼리로 충돌을 처리한다. 쿼리는 대부분의 경우 즉시 실행되므로 쿼리를 이용한 충돌 패턴에 따라 다르게 분기하여 CCT 생성시 등록된 콜백함수를 호출하여 실행한다.

21. 03. 25 추가

void ComputeInteractionsCCT(float elapsedTime);

//캐릭터 컨트롤러(CCT) 충돌 처리 시점을

//여러 개의 캐릭터 컨트롤러를 사용할 때 FixedUpdate에서 UpdatePositionCCT 호출 전 시작시 매 프레임 마다 한번 호출해 주고 사용하면 좋음

사용 예시