

PhysX :

A Practical introduction to Rigid Body Dynamics

2007.12.9

김성익 (noerror@hitel.net)

NTREEVSOFT

Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

Overview

- PhysX SDK 소개
- PhysX 구성 요소 소개
- 기본 샘플 예제를 이용한 구성 요소 Review
- 구성 요소 정리

- 오브젝트 바인딩
- 이슈



Hand

PhysX SDK 소개

- 물리 엔진
- by AGEIA (<http://www.ageia.com>)
- 라이선스
 - PC용 SDK package : 무료
 - 풀 소스 라이선스 (xbox 360, pc) : 50,000\$

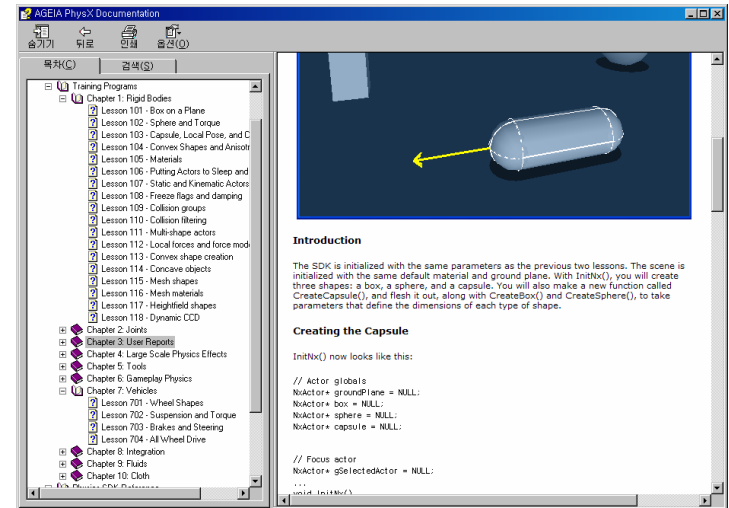
PhysX™
by ageia

Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

PhysX SDK 특징(1)

- 무료
- 충실한 튜토리얼 샘플제공
- 지속적인 업데이트
- 빠른 물리 연산
- 심플한 프로그래밍 인터페이스
- 드라이버를 반드시 설치해야 됨
- 소스 코드는 별도의 라이선스 필요



PhysX SDK 특징(2)

- 복잡한 강체 물리 시스템
- 캐릭터 컨트롤
- 자동차 다이내믹스
- 멀티 스레드/멀티 플랫폼/PPU지원
- 유체 시뮬레이션
- 옷, 깃발 등의 천 시뮬레이션
- 소프트 바디
- 역장 시뮬레이션

PhysX 구성 요소

- Scene
- Actor
 - Shape/Multi shape
 - Material
 - Force
- Joint

- Framework
- Thread Model
- Remote Debugger

기본 흐름

- SDK 초기화
- SDK 파라미터 세팅
- Scene 생성

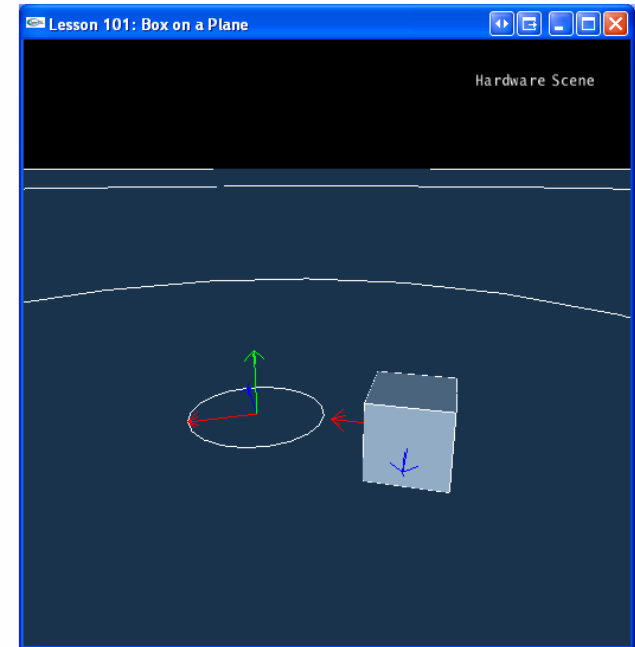
- Scene 에 머트리얼 추가
- Actor 추가
 - 위치, COM (center of mass), 머트리얼, Mass(혹은 Density) 등 설정
 - Shape
- Joint 추가

- Update

- 정보 얻어서 렌더링

샘플 Review(1) : Box on plane

- 간단한 Box를 이용한 Rigid Body Dynamics 샘플
- 처음 접해보기 좋음
- (동영상1)



샘플 Review(2)

- SDK 초기화

```
// Create the physics SDK
gPhysicsSDK = NxCreatePhysicsSDK(NX_PHYSICS_SDK_VERSION);
if (!gPhysicsSDK) return;

// Set the physics parameters
gPhysicsSDK->setParameter(NX_SKIN_WIDTH, 0.01);

// Set the debug visualization parameters
gPhysicsSDK->setParameter(NX_VISUALIZATION_SCALE, 1);
gPhysicsSDK->setParameter(NX_VISUALIZE_COLLISION_SHAPES, 1);
gPhysicsSDK->setParameter(NX_VISUALIZE_ACTOR_AXES, 1);
```

- SCENE 생성 및 초기화

```
// Create the scene
NxSceneDesc sceneDesc;
sceneDesc.simType          = NX_SIMULATION_HW;
sceneDesc.gravity          = gDefaultGravity;
gScene = gPhysicsSDK->createScene(sceneDesc);
if (!gScene){
    sceneDesc.simType          = NX_SIMULATION_SW;
    gScene = gPhysicsSDK->createScene(sceneDesc);
    if (!gScene) return;
}
```

샘플 Review(3)

- 머트리얼 설정

```
// Create the default material
NxMaterial* defaultMaterial = gScene->getMaterialFromIndex(0);
defaultMaterial->setRestitution(0.5);
defaultMaterial->setStaticFriction(0.5);
defaultMaterial->setDynamicFriction(0.5);
```

- 박스 Actor 생성

```
NxActor* CreateBox( )
{
    // Set the box starting height to 3.5m so box starts off falling onto the ground
    NxReal boxStartHeight = 3.5;

    // Add a single-shape actor to the scene
    NxActorDesc actorDesc;
    NxBodyDesc bodyDesc;

    // The actor has one shape, a box, 1m on a side
    NxBoxShapeDesc boxDesc;
    boxDesc.dimensions.set(0.5,0.5,0.5);
    actorDesc.shapes.pushBack(&boxDesc);

    actorDesc.body = &bodyDesc;
    actorDesc.density = 10;
    actorDesc.globalPose.t = NxVec3(0,boxStartHeight,0);
    return gScene->createActor(actorDesc);
}
```

샘플 Review(4)

- 업데이트

```
void StartPhysics()  
{  
    // Update the time step  
    gDeltaTime = UpdateTime();  
  
    // Start collision and dynamics for delta time since the last frame  
    gScene->simulate(gDeltaTime);  
    gScene->flushStream();  
}  
  
void GetPhysicsResults()  
{  
    // Get results from gScene->simulate(gDeltaTime)  
    while (!gScene->fetchResults(NX_RIGID_BODY_FINISHED, false));  
}
```

- 힘

```
NxVec3 ApplyForceToActor(NxActor* actor, const NxVec3& forceDir, const NxReal forceStrength)  
{  
    NxVec3 forceVec = forceStrength*forceDir*gDeltaTime;  
    actor->addForce(forceVec);  
    return forceVec;  
}
```

샘플 Review(5)

- 결과 얻기 / 적용

```
NxShape*const shapes = actor->getShapes();
NxU32 nShapes = actor->getNbShapes();
while (nShapes-->0)
{
    DrawShape(shapes[nShapes], useShapeUserData);
}
```

```
void DrawBox(NxShape* box)
{
    NxMat34 pose = box->getGlobalPose();

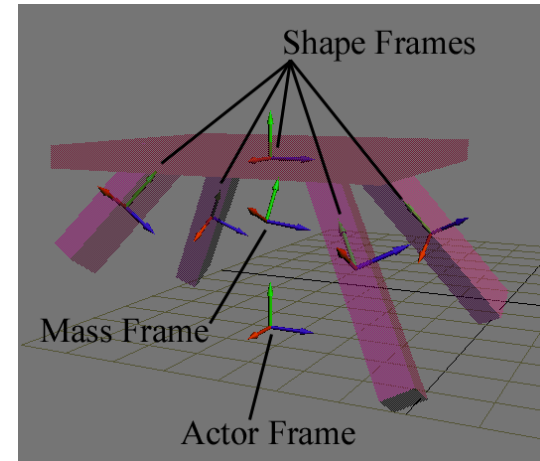
    glPushMatrix();
    SetupGLMatrix(pose.t, pose.M);
    NxVec3 boxDim = box->isBox()->getDimensions();
    glScalef(boxDim.x, boxDim.y, boxDim.z);
    RenderBox();
    glPopMatrix();
}
```

Scene

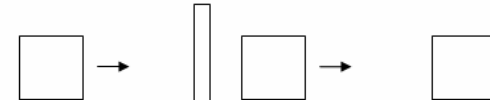
- 시뮬레이션이 이루어지는 Scene
- NxPhysicsSDK 객체를 이용해서 생성, 소멸
 - NxPhysicsSDK ::createScene, NxPhysicsSDK::releaseSDK
- Multiple Scene
- Actor, Joint, Material 등 물리 객체 생성 소멸 관리
 - Actor : 물리 객체
 - Joint : 물리 객체를 구속
 - Material : 물리 객체의 표면 (마찰, 탄성)

Actor(1)

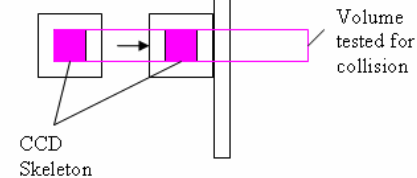
- 물리 객체
- 생성
 - Shape
 - Plane, Sphere, Capsule, Box
 - Heightfield
 - ConvexhullMesh
 - TriangleMesh
 - Multi Shape
 - CCD
- 속성
 - 무게, 관성
 - 위치, 모멘트
 - 속도, 각속도
 - 힘, 토크



Without CCD



With CCD

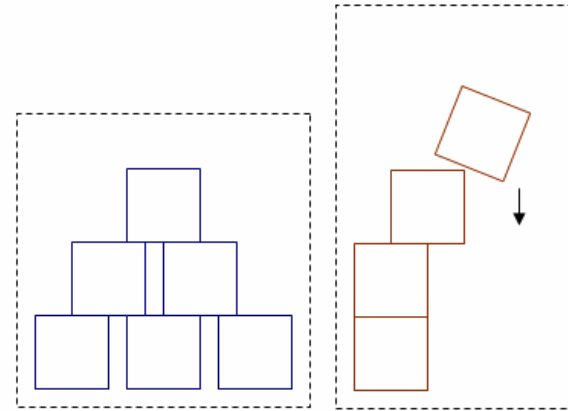


Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

Actor(2)

- 기타 속성
 - 충돌 그룹
 - gravity, frozen, kinematic, sleep
- Awake/Sleep
- Sweep



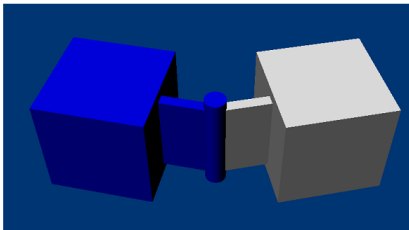
	NxSphereShape	NxBoxShape	NxCapsuleShape	NxPlaneShape	NxConvexShape	NxHeightfieldShape	NxWheelShape	NxTriangleMeshShape	NxTriangleMeshShape (Heightfield)	NxTriangleMeshShape (pmap)
NxSphereShape	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NxBoxShape	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NxCapsuleShape	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NxPlaneShape	✓	✓	✓	✗	✓	✗	✓	✓	✗	✗
NxConvexShape	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NxHeightfieldShape	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗
NxWheelShape	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
NxTriangleMeshShape	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗
NxTriangleMeshShape (heightfield)	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗
NxTriangleMeshShape (pmap)	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓

need

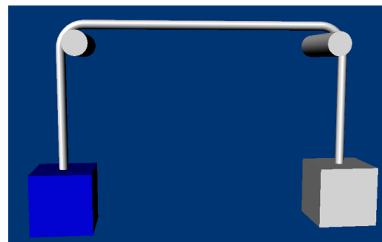
Joint

- Actor를 구속 (축속, 각도, 거리, 위치)
- 종류
 - 6 DOF. Revolute, Pulley : 모터
 - 6 DOF, Spherical, Revolute, Distance : 스프링
 - 6 DOF, Spherical, Revolute : 각 제한
 - Prismatic, Fixed, Point In Plane, Point On Line
- 부서지는 Joint

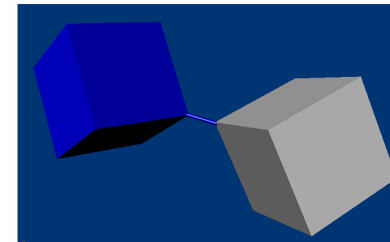
Revolute Joint



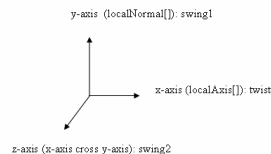
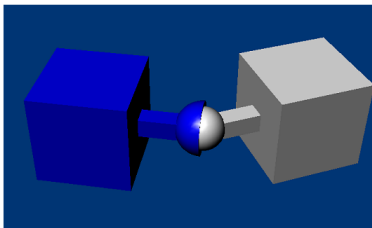
Pulley Joint



Distance Joint

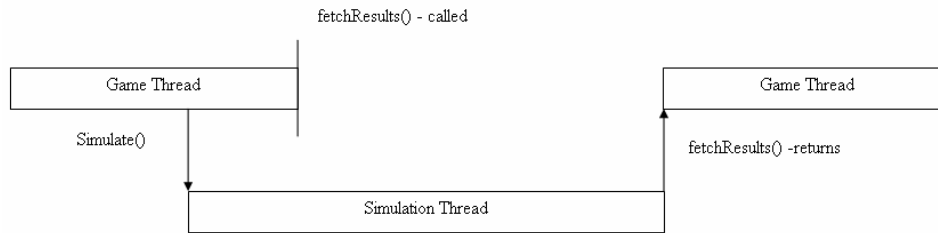


Spherical Joint



Thread model

- 기본적으로 내부 스레드로 물리연산
 - NX_SF_SIMULATE_SEPARATE_THREAD



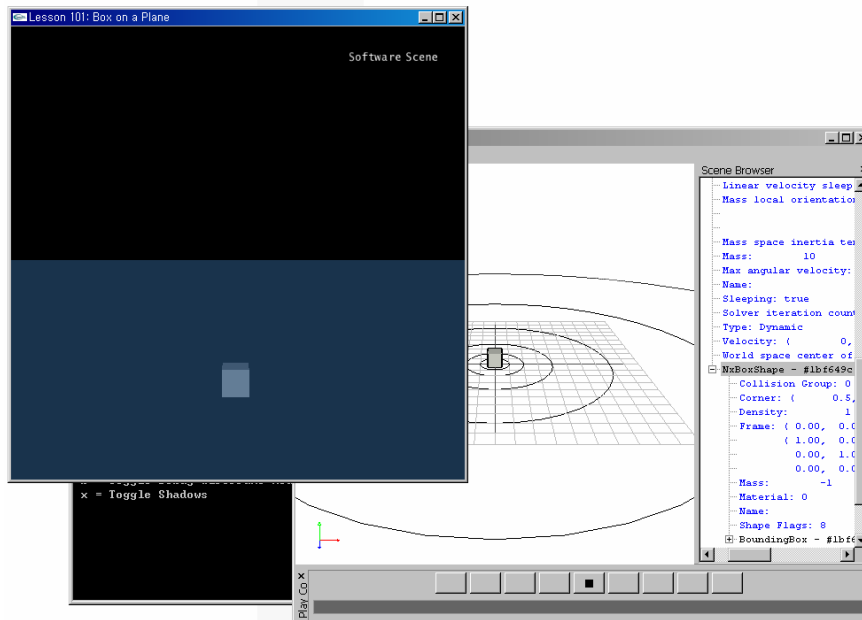
- 백그라운드 작업 중 Scene, Actor 객체 접근

Remote Debugger

- 어플리케이션 독립적으로 씬 디버깅

```
// Create the remote debugger
gRemoteDebugger = gPhysicsSDK->getFoundationSDK().getRemoteDebugger();
gRemoteDebugger->connect("localhost");
```

- 리플레이
- (동영상2)



Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

좌표계

- 회전과 이동 분리
- Direct3D 좌표계

```
void SetGlobalTM(NxActor *actor, const D3DXMATRIX & mat)
{
    NxMat33 tm;
    tm.setColumnMajorStride4(&mat._11);

    actor->setGlobalOrientation(tm);
    actor->setGlobalPosition(NxVec3(mat._41, mat._42, mat._43));
}

void GetGlobalTM(NxShape *shape, D3DXMATRIX & mat)
{
    NxMat34 pose = shape->getGlobalPose();

    pose.M.setColumnMajorStride4(&mat._11);
    mat._14 = mat._24 = mat._34 = 0;

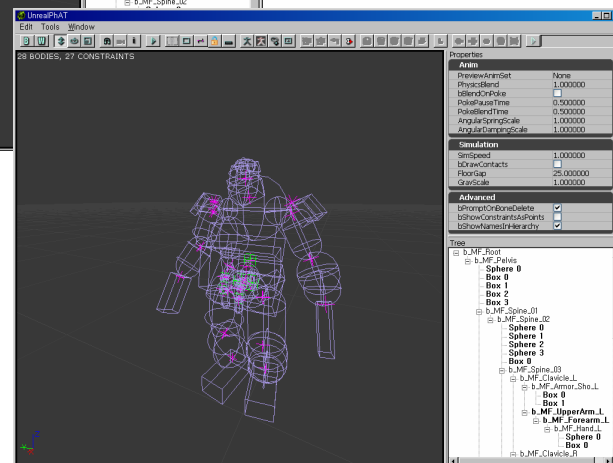
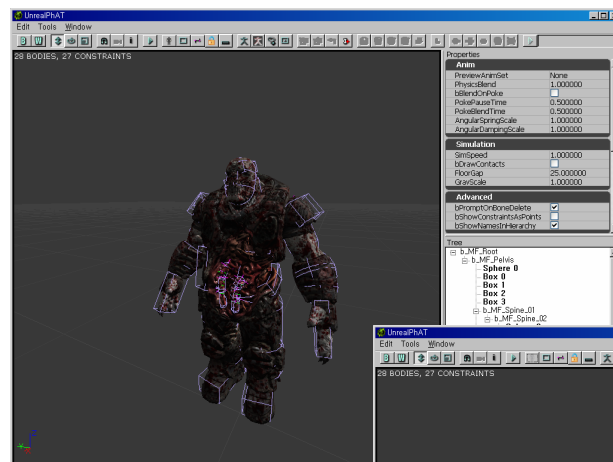
    mat._41 = pose.t.x;
    mat._42 = pose.t.y;
    mat._43 = pose.t.z;
    mat._44 = 1;
}
```

물리 모델링

- 단순화
- 단순화
- 단순화

- (동영상3)

- Joint 활용한 계층적인 구조



Spirit of Flame
3D RealTime Graphics Programming Study

Kasa

바인딩(1)

- 좌표계
- Actor TM => 오브젝트 TM
- 오브젝트와 Actor 초기위치가 동일 X
 - Actor 초기 TM = 오브젝트 TM * PivotTM
 - Actor world TM * Inverse(PivotTM) => 오브젝트 TM

바인딩(2)

- 계층 구조 오브젝트

- $A \leftarrow B$

- $A^{TM} = A^{LOCAL} * TM^{WORLD}$

- $B^{TM} = B^{LOCAL} * A^{TM}$

- $B^{LOCAL} = B^{TM} * Inverse(A^{TM})$

- A', B' Actor와 A'와 B'를 Joint로 연결

- $A'^{TM} = A'^{LOCAL} * TM^{WORLD}$

- $B'^{TM} = B'^{LOCAL} * A'^{TM}$

- $B'^{LOCAL} = B'^{TM} * Inverse(A'^{TM})$

- Actor 의 WORLD TM을 이용해서 Object의 Local TM을 구함

- Rag Doll

바인딩(3)

- 단순화된 계층 구조 오브젝트
 - $A \leftarrow B \leftarrow C \leftarrow D$ 를 $(A' \leftarrow B' \leftarrow C') \leftarrow D'$ 로 모델링
 - Actor 2개와 1개의 조인트로 구성
 - B^{LOCAL} , C^{LOCAL} 은 고정
 - A의 TM은 $(A' \leftarrow B' \leftarrow C')$ 액터로 결정
 - D의 local TM은 $(A' \leftarrow B' \leftarrow C')$ 와 D' 액터로 결정
 - $D^{TM} = D^{LOCAL} * C^{LOCAL} * B^{LOCAL} * A^{TM}$
 - $D'^{TM} = D^{LOCAL} * C^{LOCAL} * B^{LOCAL} * (A' \leftarrow B' \leftarrow C')^{TM}$
 - $D^{LOCAL} = D'^{TM} * \text{Inverse}(C^{LOCAL} * B^{LOCAL} * (A' \leftarrow B' \leftarrow C')^{TM})$

이슈

- 퍼포먼스
- 래그 돌

- 엔진 통합
- 물리 추상 객체

- 서버 사이드 다이내믹스

- PPU

질문

-

Spirit of Flame
3D RealTime Graphics Programming Study

Kasa