1. Lógica del Disparo

Para la implementación del disparo hemos usado el objeto ***ClosestRayResultCallback*** como callback en la invocación al método ***rayTest***() dado por ***Bullet***.

Con éste método conseguimos hacer una consulta de aquel objeto con el que hemos colisionado si lanzamos un rayo desde un punto A a un punto B. Recibiendo en el callback ésta información, es decir, si hemos colisionado con algo y si lo hemos hecho nos diría con que objeto ha sido. Aquí detallamos la parte del código encargada de la detección de colisiones.

// Start and End are vectors

btCollisionWorld::ClosestRayResultCallback rayCallback(posIni, posFin);

// Perform raycast

world->getBulletDynamicsWorld ()->rayTest(posIni, posFin, rayCallback);

if(rayCallback.hasHit())

{

btCollisionObject\* obA = (btCollisionObject\*)(rayCallback.m\_collisionObject);

OgreBulletCollisions::Object \*obOB\_A = world->findObject(obA);

int sizeCharacteres = characteres.size();

Character \*character = NULL;

if (obOB\_A != getRigidBody())

{

for (int i = 0; i < sizeCharacteres && (\*characterCollision) == NULL; i++)

{

character = characteres[i];

if (obOB\_A == character->getRigidBody() && character->\_stateCaracter == LIVE)

{

(\*characterCollision) = character;

}

}

}

isCollition = true;

}

Para el aspecto visual del disparo hemos empleado un objeto manual de OGRE (***createManualObject***) dando dos puntos y la linea que los une, nos permite dar la sensación de estela en el disparo. Aquí adjuntamos las lineas de código encargadas de la creación de ésta:

void Utilities::put\_shot\_in\_scene ( Ogre::SceneManager\* sceneMgr, string name, Ogre::SceneNode\* nodeShot, Ogre::ManualObject\*\* shot)

{

\*shot = sceneMgr->createManualObject(name + "LineSHOT");

(\*shot)->begin("MaterialShot", Ogre::RenderOperation::OT\_LINE\_LIST);

(\*shot)->position(Ogre::Vector3::ZERO);

Ogre::Vector3 posFin(nodeShot->getOrientation().zAxis() \* 1);

(\*shot)->position(posFin);

(\*shot)->end();

(\*shot)->setCastShadows(true);

nodeShot->attachObject ( (\*shot) );

}