PHI: Physics Application Programming Interface

Bing Tang, Zhigeng Pan, ZuoYan Lin, Le Zheng

State Key Lab of CAD&CG, Zhejiang University,
Hang Zhou, China, 310027
{btang, zgpan, linzouyan, zhengle}@cad.zju.edu.cn

Abstract. In this paper, we propose to design an easy to use physics application programming interface (PHI) with support for pluggable physics library. The goal is to create physically realistic 3D graphics environments and integrate real-time physics simulation into games seamlessly with advanced features, such as interactive character simulation and vehicle dynamics. The actual implementation of the simulation was designed to be independent, interchangeable and separated from the user interface of the API. We demonstrate the utility of the middleware by simulating versatile vehicle dynamics and generating quality reactive human motions.

1 Introduction

Each year games become more realistic visually. Current generation graphics cards can produce amazing high-quality visual effects. But visual realism is only half the battle. Physical realism is another half [1]. The impressive capabilities of the latest generation of video game hardware have raised our expectations of not only how digital characters look, but also they behavior [2].

As the speed of the video game hardware increases and the algorithms get refined, physics is expected to play a more prominent role in video games. The long-awaited Half-life 2 impressed the players deeply for the amazing Havok physics engine[3]. The incredible physics engine of the game makes the whole game world believable and natural. Items thrown across a room will hit other objects, which will then react in a very convincing way. Even dead bodies that are close to the radius of explosion will be tossed around like rag-doll.

However, there are a number of open issues that still to be addressed especially regarding the integration of physics algorithms with the rest of the graphics system in a game engine. Game developers have to spend lots of time to integrate them into their games. Most physics engines in today’s game are pretty weak, especially for articulated character animation. The characters are only dealt with limp and dead rag-doll in physics simulation. Rag-doll simulation appears lifeless because it lacks a control system to generate human behavior[4]. In this paper, we propose an independent physics middleware to incorporate physics into games. We also introduce a new method for simulating reactive motions for data-driven animation. The goal is to generate realistic behaviors under unexpected external forces. A set of techniques are introduced to select a motion capture sequence which follows an impact, and then synthesize a believable transition to this found clip for character interaction.

2 Previous work

In the past ten years, physics has received much attention to improve the realism of virtual environments. Looking through the dynamics literature, a large variety of dynamics formulation can be found, such as Newton-Euler, Gibbs-Appel and Gauss' Least Constraint Principle.

Some software libraries have been developed to perform rigid body simulation. Commercial dynamics libraries such as Havok [3] and PhysX [5] have already been deployed into popular commercial games such as Half Life 2, Max Payne 2, and the future Unreal 3 engine technology. There are also several freely available libraries with comparably robust features. The Open Dynamics Engine (ODE) [6] is a free, industrial quality library for simulating articulated rigid body dynamics. Tokamak [7] is another library freely available, excelling in efficient joint constraints and stacking large number of objects. However, these physics libraries are not designed well to combine with existing applications. It is a challenging work to integrate the physics engines into games.

The most closely related work is Open dynamics framework (ODF)[8]. ODF is an open source project that provides tools to make it easier to get dynamics content into games. These tools are provided as sample code, scripts, and plug-in components that can be integrated directly into an existing tools pipeline or into major digital content creation tools such as 3D Studio Max, Maya, or SoftImage. Unfortunately, ODF is against its primary goal to support various physics engines and becomes a demo application on top of PhysX. Conversely, our goal is to provide an independent physics middleware to incorporate physics into games.

3 System Framework

The independent physics middleware was archived by dividing the API in three layers: wrapper, communication, and implementation. The main work was to define an appropriately abstract physical descriptor, then define and implement all sub-descriptors for every concrete application programming interface. First we bound the wrapper layer with low level graphics API of Open GL and DirectX 3D to test the stability. Then we replaced low level graphics libraries with high level game engine (such as OGRE and Irricht) and improved the common interface. PHI was written in C++ and the supported low level physics libraries are at this time ODE and PhysX. PHI encapsulated almost every feature of these physics engines, and added higher level features such as per shape material setting, vehicle dynamics configuration and XML physics script data loading. The physics data allows developer to do quick prototyping, and an exporter that creates rigid bodies with appropriate bounding shapes directly from source artwork.

4 Implementation

PHI contains powerful customizable modules for the user to set up various applications. We implemented two set of examples to demonstrate the feasibility of the physics middleware: dynamics vehicle simulation and reactive human motions.

4.1 Vehicle simulation

PHI contains a powerful customizable vehicle module. Users can create cars, trucks or tanks from a set of vehicle specific scripts. Benefit from PHI, vehicle simulation combines physics with graphics well. It is easy to construct a well-looking jeep or armored car running on terrains with full dynamics by PHI (shown as Fig 1). These vehicles are presented as articulated structure of a series of body parts connected by joints, and they driven by motor joints under dynamics. When a large force acts on the vehicles, the joints would break to respond destructive results. We use four groups of spring-damper to simulate the vehicle's suspension and find they work well. Features currently included are the dynamic behaviour of the driving, suspension, steering and firing. The user can create parameterized vehicles which can be driven interactively in a Virtual Environment (VE).

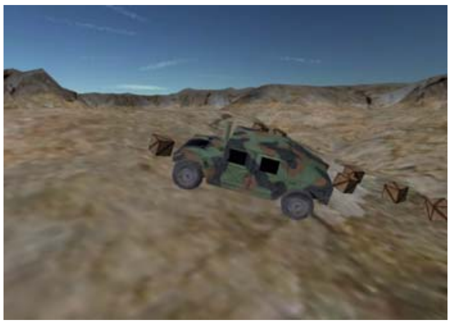
Fig. 1. Vehicle Simulation

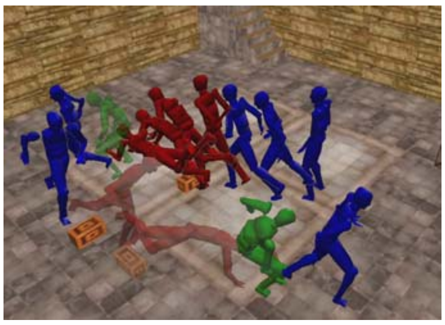
Fig. 2. Reactive human motion generation

4.2 Reactive human motions generation

It is very important to create live and vivid animal or biped character in today's game. Simulation human motion with realistic responses to unexpected impacts in a controllable way is challenging work in motion synthesis.

With PHI, we proposed a novel method for incorporating unexpected impacts into a motion capture animation through physical simulation. Fig 2 shows a range of responses when a character falls down for encountering unexpected obstacles. During the fall process, arms of the character are controlled to generate convincing human behaviors: arms are rapidly elevated backward and outward in an attempt to stabilize the forward displaced COM after collision, and then adjusted to track the predicted landing location of the body and generate protective behaviors to avoid injury. The attached video demonstrates our results using real-time screen capture where user interacts with the character.

5 Conclusion and future work

An easy to use middleware for integrating physics simulation into games has been successfully designed and tested. Both the demo and videos of implementation can be downloaded from the world-wide web at http://www.cad.zju.edu.cn/vrmm/PHI. Using PHI to improve the physical realism of virtual world proved to be effective and robust. In the future work, we will make it possible for PHI with the support of more graphics engines and physics engines. In addition, Collada support will be implemented to aid developers.

Acknowledgements

This research work is co-founded under Microsoft Research Asia (MSRA) Regional Theme Project 2005, Project 973 (grant NO: 2002CB312100) and NSFC (grant NO: 60533080).

References

1. Eberly, D.H., Shoemaker, K.,Shoemake, K.: Game Physics. Morgan Kaufmann Pub. (2003) 221-361
2. Kokkevis, E.: Practical Physics for Articulated Characters. Game Developers Conference 2004, (2004)
3. Reynolds, H.,Collins, S.: Havok 2: Game dynamics sdk.(2005).http://www.havok.com
4. Zordan, V.B., et al.: Dynamic response for motion capture animation. ACM SIGGRAPH 2005. ACM Press, (2005) 697-701
5. Morav'anszky, A.: PhysX SDK. NovodeX AG,(2005).http://www.ageia.com/novodex.html
6. Smith, R.: Open dynamics engine. 2005).http://www.ode.org
7. Lam, D.: Tokamak game physics sdk. 2005).http://www.tokamakphysics.com
8. Ratcliff, J.W.: Open dynamics framework. 2005).http://www.physicstools.org