# ВИЗУАЛИЗАЦИЯ ЗАДАЧ ПО КУРСУ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА»

**О. Г. Казанцева, Е. А. Корчевская**

*Витебский государственный университет имени П. М. Машерова*
*Витебск, Беларусь*
*E-mail:* kazantsevaog@vsu.by, myaumur@tut.by

В работе рассмотрены возможности графических библиотек Newton Game Dynamics и OpenGL для наглядного представления динамики механических процессов.

*Ключевые слова*: графическая библиотека, механическая система.

Хорошее усвоение курса «Теоретическая механика» требует глубокого изучения теории, приобретения твердых навыков в решении задач. Для этого необходимо иметь четкое представление о поведении механической системы.

При изучении курса «Теоретическая механика» студенты часто сталкиваются со сложностями визуального характера: невозможностью наглядно представить механическую систему в начальный момент времени, непониманием динамики движения объектов. Как следствие этого студенты затрудняются в указании верных направлений составляющих скорости и ускорения. Чтобы избежать таких проблем, необходимо дать студентам возможность увидеть изображение механической системы в любой момент времени для конкретной задачи. Для этого можно использовать различные графические пакеты.

Графические пакеты широко используются в игровой индустрии и профессиональных трехмерных редакторах. Очевидно, что они могут быть использованы для моделирования различных объектов при изучении курса «Теоретическая механика».

В данной работе мы рассматриваем использование библиотек Newton Game Dynamics и OpenGL для визуализации учебных задач.

Newton Game Dynamics является встраиваемой библиотекой для моделирования физических процессов в реальном времени.

Библиотека Newton Game Dynamics легко интегрирутся в любое приложение. В этой технологии необходимо знать только основные физические принципы, чтобы реализовать реалистичное поведение объектов.

К возможностям библиотеки относятся: моделирование динамики твердого тела, определение коллизий, моделирование динамики сложных объектов и многое другое. Большим достоинством является то, что использование Newton Game Dynamics позволяет пользователям создавать собственные приложения, моделирующие динамику механических процессов.

Для визуализации объектов используется графическая библиотека OpenGL с использованием вспомогательной библиотеки Glut.

## ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ С БИБЛИОТЕКОЙ NEWTON

Первым делом необходимо подключить заголовочный модуль Newton.h и библиотеку Newton.lib к проекту. Данные файлы идут в стандартной поставке Newton Game Dynamics SDK [1].

Далее необходимо инициализировать саму библиотеку.

Основными понятиями библиотеки являются твердые тела (rigid body), далее – просто тело. Тела характеризуются массой, скоростью и ускорением (угловыми и линейными). К телам можно прикладывать силы, вращающий момент, добавлять импульс.

Объекты коллизии могут быть различной формы: параллелепипед, эллипсоид, конус, капсула, цилиндр, а также пустой объект. Также имеется возможность составлять комбинированные объекты.

По умолчанию центр масс тела находится в точке (0, 0, 0) относительно тела, т. е. в центре тела. Обычно это положение дает приемлемые по точности результаты моделирования, однако иногда встречаются ситуации, когда центр масс необходимо сместить, например: тело имеет асимметрию или для каких-либо специальных эффектов.

Основными функциями библиотеки для работы с твердыми телами являются:

- задание массы тела и момента инерции относительно каждой из осей;
- получение информации о массе тела;
- определение положения тела в пространстве;
- получение текущей матрицы трансформации тела;
- смещение центра масс;
- задание общей линейной и угловой скорости.

Библиотека дает наиболее точные результаты, если массы самого тяжелого и самого легкого объекта различаются не более чем на 200 кг.

Newton Game Dynamics спроектирована таким образом, чтобы быть как можно более независимой по отношению к приложению. Она создавалась для того, чтобы обрабатывать массивы данных порядка тысячи тел, не заботясь о том, как они представляются в приложении. Поэтому когда создается новое тело, имеется возможность задавать пользовательские данные, а также несколько указателей на т. н. функции-события (callback functions). Newton использует эти функции для того, чтобы связываться с приложением всякий раз, когда поменяется какой-либо параметр тела.

Опишем некоторые функции события, применимые для твердых тел.

- Destructor callback – вызывается перед уничтожением тела.
- Force and torque callback – может использоваться для добавления различных сил и вращающего момента к телу. Это может быть, например, сила притяжения или другая внешняя сила.
- Transformation callback – вызывается каждый раз, когда тело поменяет свое положение в пространстве. Если, например, с телом связан геометрический примитив, то основное приложение может не заботиться о том, когда объект поменял свое положение. Все это сделает библиотека.

Ограничения (constraints) или joint'ы – это объекты, которые соединяют два тела, ограничивающие взаимное перемещение тел относительно друг друга по одному или более направлениям. Newton API оперирует рядом различных joint'ов: шарнирное соединение (ball-socket joint), петельное соединение (hinge joint), слайдер (slider joint), штопор (corkscrew joint). Имеется возможность комбинировать различные виды соединений, а также создавать пользовательские типы.

## ПРИМЕР ВИЗУАЛИЗАЦИИ ЗАДАЧИ

В качестве примера рассмотрим следующую задачу [2].

Точка $M$ движется вдоль прямой $OA$ со скоростью $U$, а сама прямая вращается в плоскости $Ox_1y_1$ вокруг центра $O$ с угловой скоростью $\omega$. Определить скорость точки $M$ относительно осей $Ox_1y_1$ в зависимости от расстояния $OM = r$.

В качестве прямой берем цилиндр и связываем его с твердым телом:

```
    cylinder = new CylinderPrimitive(nGrapphicWorld, location,
0.1f,5.0f);
    collision = NewtonCreateCylinder(nWorld,0.1f,5.0f, NULL);
    boxBody1 = NewtonCreateBody (nWorld, collision);
    NewtonBodySetUserData (boxBody1, cylinder);
    NewtonBodySetTransformCallback (boxBody1,
PhysicsSetTransform);
```

Помещаем на него еще одно тело сферической формы и связываем с цилиндром посредством связи слайдер:

```
    collision = NewtonCre-
ateSphere(nWorld,0.15f,0.15f,0.15f,NULL);
    boxBody2 = NewtonCreateBody (nWorld, collision);
    location.m_posit.m_x=-9.0f;
    sphere = new SpherePrimi-
tive(nGrapphicWorld,location,0.15f,0.15f,0.15f);
    sphere->userdata = 1;
    NewtonBodySetUserData (boxBody2, sphere);
    NewtonBodySetTransformCallback (boxBody2, PhysicsSetTrans-
form);

    dVector pin(1,0,0);
    joint = NewtonConstraintCreateSlider(nWorld,
&location.m_posit.m_x, &pin.m_x,boxBody1,boxBody2);
```

У цилиндра смещаем центр масс в левый конец и придаем ему угловую скорость:

```
    dVector pivot(-2.5,0,0);
    NewtonBodySetCentreOfMass(boxBody1,&pivot.m_x);
    pivot.m_x=0;
    pivot.m_y=0;
    pivot.m_z=0.1;
    NewtonBodySetOmega(boxBody1,&pivot.m_x);
```

Сфере придаем линейную скорость:

```
    pivot.m_z = 0;
    pivot.m_x = 0.1;
    NewtonBodySetVelocity(boxBody2,&pivot.m_x);
```

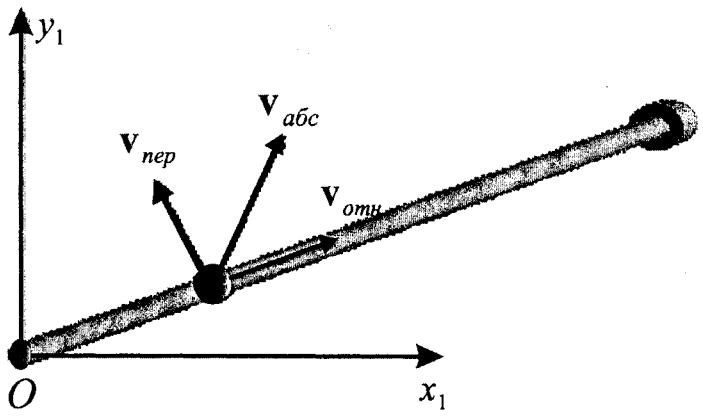


*Рис. 1.* Движение точки по вращающейся прямой в момент времени 1

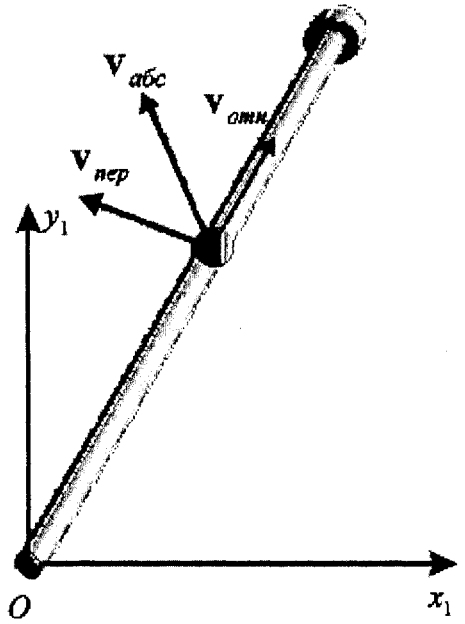


*Рис. 2.* Движение точки по вращающейся прямой в момент времени 2

## ЛИТЕРАТУРА

1. Newton Game Dynamics [Electronic resouce]. – Mode of access : www.physicsengine.com.
2. *Корчевская, Е. А.* Теоретическая механика: кинематика и динамика материальной точки / Е. А. Корчевская, Г. И. Михасев. – Витебск : ВГУ, 2005.