электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 681.527

Использование Irrlicht Engine для визуализации работы лабораторных стендов по исследованию КПД редукторов

***Бошляков И.А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Специальная Робототехника и Мехатроника»
***Коновалов К.В.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Специальная Робототехника и Мехатроника»
***Метасов И. Е.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Специальная Робототехника и Мехатроника»
***Шереужев М. А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Специальная Робототехника и Мехатроника»

Научный руководитель: Перминова Е.А., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Элементы приборных устройств»

Научный руководитель: Бошляков А.А.,к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Специальная Робототехника и Мехатроника»
kafsm7@sm.bmstu.ru

Введение

На кафедре РЛ5 студенты проходят лабораторные работы на темы «Исследование КПД червячного и планетарного редукторов». Реальные стенды (Рис. 1,2) необходимо обслуживать, они находятся в эксплуатации долгое время, и периодически выходят из строя, что требует более частого обслуживания. В связи с этим, есть необходимость разработать виртуальные модели стендов, и проводить лабораторные работы на компьютере. Поэтому была поставлена задача разработки компьютерной модели реальной лабораторной установки и на ее основе комплексной мультимедийной программы всей лабораторной работы.

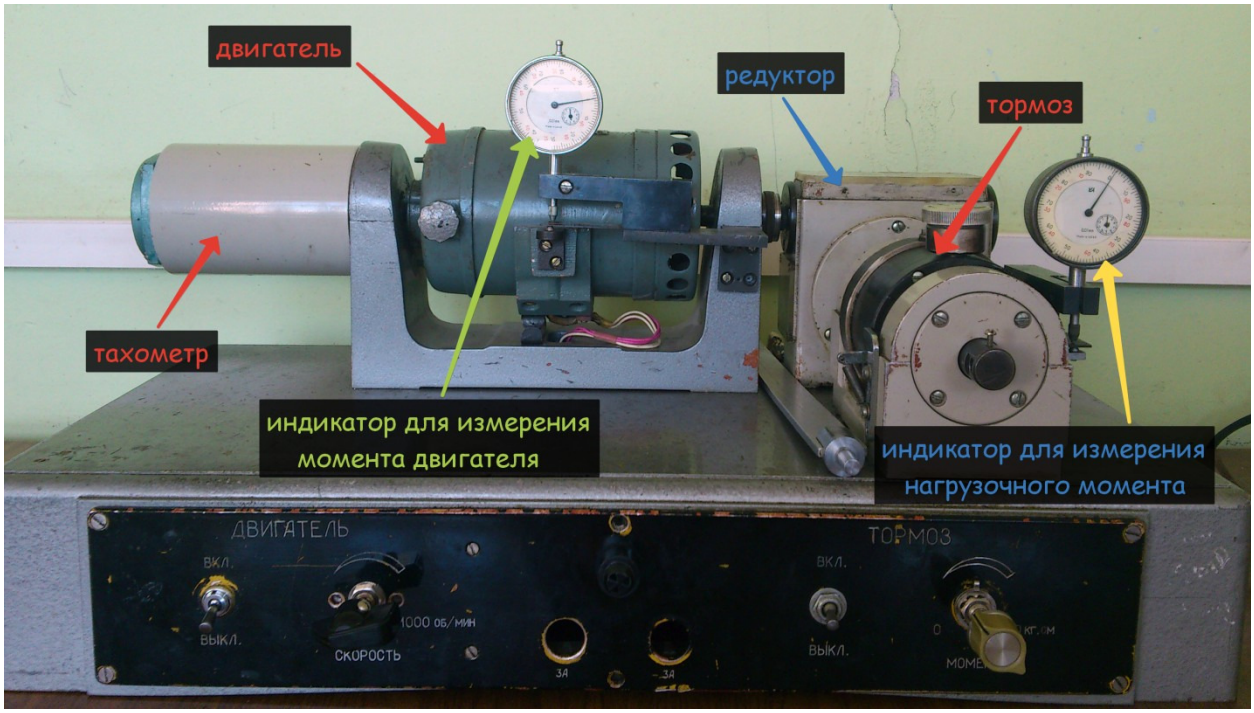

Рис. 1. Реальный стенд червячной передачи

Рис. 2. Реальный стенд планетарной передачи

Данная задача уже решалась с использованием пакета *MatLab*, и рассмотрена в статье Д. Г. Остапенко, И. А. Мурзин «Моделирование лабораторной работы по курсу ОКП «Исследование КПД прямозубого зубчатого редуктора», но данный метод требует очень больших ресурсов от компьютера. Для решения этой проблемы предлагается использовался другой метод с использованием пакета МВТУ, *SolidWorks* и графического

движка Irrlicht Engine, который управлялся с помощью МВТУ. Основное внимание в данной статье уделено визуализации управляемой *3D* модели.

Особенности работы с *Irrlicht Engine*.

Irrlicht (*Irrlicht Engine*) — трёхмерный графический движок, высоко производительный, написанный на C++, который является бесплатным свободным программным продуктом и распространяется на условиях лицензии *zlib*. Содержит в себе мощное высокоуровневое *API* для создания *2D* и *3D* приложений, таких как прикладная визуализация.

Irrlicht использует возможности *OpenGL*, *DirectX* и нескольких собственных рендереров. Пользователю предоставляются различные функциональные возможности по загрузке и управлению трёхмерными (*3D*) объектами (сцены, модели и т. п.), немногими спецэффектами и графическим интерфейсом пользователя. Не требует подключения сторонних модулей для реализации высокоуровневых функций (есть простейшая физика, *GUI* (графический интерфейс пользователя) и т. д.). Существует три официальных дополнения для *Irrlicht*: *IrrKlang* (аудиобиблиотека), *IrrXML* (загрузка и обработка *XML*-файлов), *IrrEdit* (редактор сцен). Для использования расширенных функций физики существует физический движок *ChronoEngine* (по причине того, что в *Irrlicht* встроена примитивная физическая система).

Рис. 3. Пример попиксельного освещения в *Irrlicht*

Одна из важных особенностей *Irrlicht* его кроссплатформенность — то есть способность работать на различных платформах. Платформонезависимая прослойка обеспечивает лёгкую портируемость (то есть перенос движка) на различные не поддерживаемые официально платформы, в частности существуют порты под android, iPhone и пр.

Главные возможности и особенности:

Высокопроизводительная *3D* визуализация посредством *Direct3D* и *OpenGL*.

Платформенная независимость, запускается на *Windows*, *Linux*, *OSX*, *Solaris* и др.

Импорт многих известных форматов моделей: *Maya* (*.obj*), 3DStudio (.3ds) и др.

Движок содержит следующие пространства имен (они же модули):

irr — содержит в себе следующие пространства имен:

core — предоставляет классы общего назначения, такие как *vectors*, *planes*, *arrays*, *lists* и т. п.

gui — содержит полезные классы для упрощения создания графического интерфейса пользователя *GUI*.

scene — в этом модуле сосредоточено управление сценой: загрузка мешей (*Mesh*), специальные узлы сцены (так как *octrees* и *billboards*).

video — в этом модуле содержатся классы для доступа к видеодрайверу. Весь 2d и 3d рендринг происходит здесь.

В *Irrlicht* основное понятие – нода. Нод – это «объект». Объектом в *Irrlicht* является почти все – и модели, и камеры, и источники света.

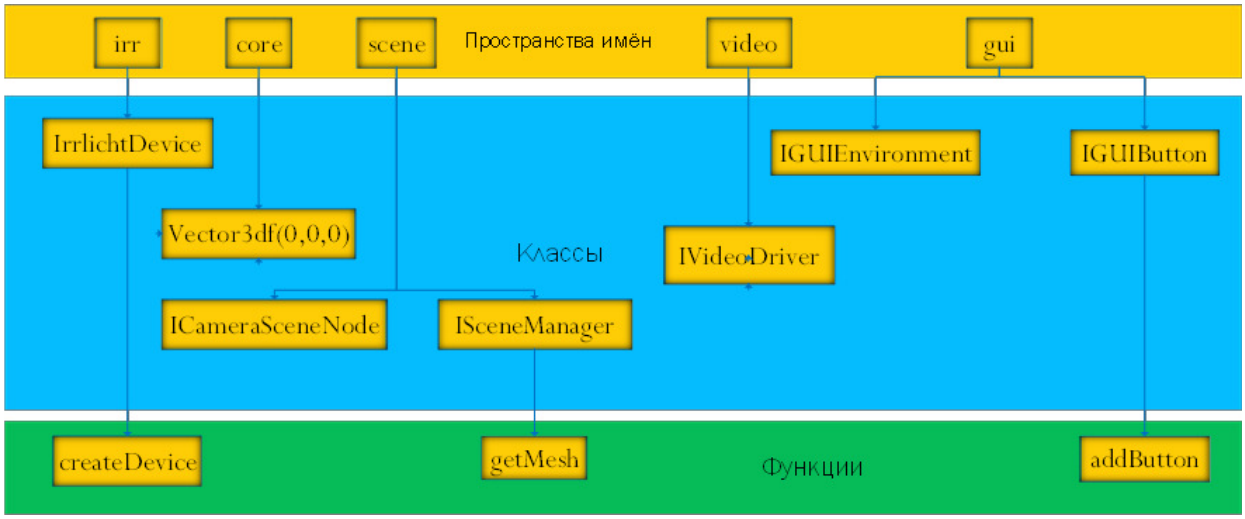

Рис. 4. Пространства имен, некоторые классы и функции *Irrlicht*

irr::IrrlichtDevice

device = createDevice(EDT_OPENGL,dimension2d<u32>(1270, 714), 32, false, false, false, 0);

Самая главная функция движка – это *createDevice()* , ею создается корневой объект движка *IrrlichtDevice* и методами которого делается много чего полезного. Функция *createDevice()* имеет 7 аргументов

deviceType: Тип устройства. Принимает значения: *EDT_SOFTWARE*, *EDT_BURNINGSVIDEO*, *EDT_NULL*, *EDT_DIRECT3D8*, *EDT_DIRECT3D9* или *EDT_OPENGL*.

windowSize: размер окна в оконном режиме или разрешение экрана в полноэкранном режиме.

bits: Количество бит для цвета на один пиксель. Может быть 16 или 32. Параметр часто игнорируется в оконном режиме.

fullscreen: определяет в оконном или полноэкранном режиме будет запущено приложение

stencilbuffer: включает трафаретный буфер (для рисования теней).

vsync: включает вертикальную синхронизацию, полезна для полноэкранного режима.

eventReceiver: объект – обработчик событий. Если в нем нет необходимости, ставить равным нулю.

irr::core::vector3df(0,0,0) задаёт точку в пространстве

irr::scene::ISceneManager этот класс отвечает за сцену, загрузку нодов.

getMesh("C:/Program Files/LW5/detali/bearing1.obj") загружает детали из указанной нами папки. Загруженные детали с помощью функций *setPosition* и *setRotation* можно задать положение и ориентацию в пространстве.

irr::scene::ICameraSceneNode позволяла добавлять различные виды камер, например *addCameraSceneNodeFPS()*, камера управляемая с помощью стрелок и мыши; *addCameraSceneNode(0,vector3df(110,50,50),vector3df(0,0,0)),* стационарная камера, которая находится в точке 110,50,50 и смотрит в точку 0,0,0.

irr::video::IVideoDriver отвечает за драйвер.

irr::gui::IGUIEnvironment отвечает за интерфейс.

irr::gui::IGUIButton позволяет добавлять кнопки с помощью функции *addButton*.

Главный цикл программы, в котором происходит вывод на экран *while(device->run()).*

Формирование сцены, освещения и создание интерфейса.

Для создания управляемых *3D* моделей стендов были написаны программы на C++ с использованием классов и функций *Irrlicht*. Сначала были разработаны модели в

SolidWorks(Рис. 5,6), затем каждая деталь по отдельности была загружена в рабочее пространство *Irrlicht*.

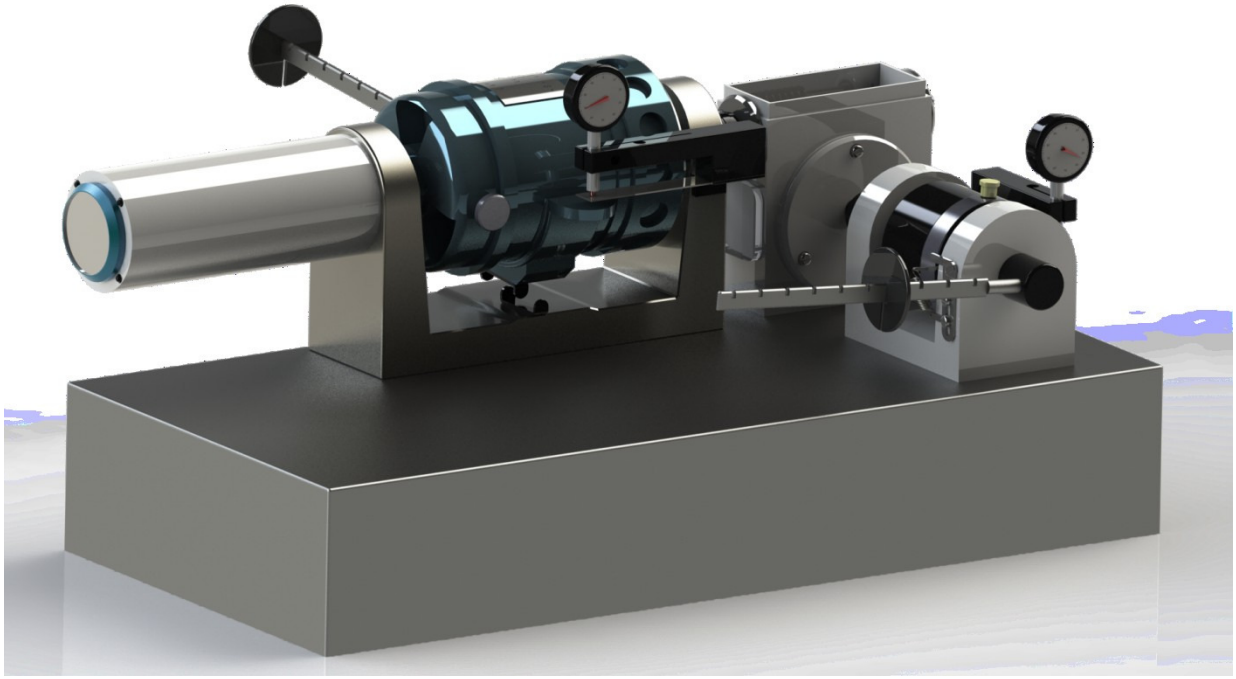

Рис. 5. Модель стенда с червячной передачей, созданной в *SolidWorks*

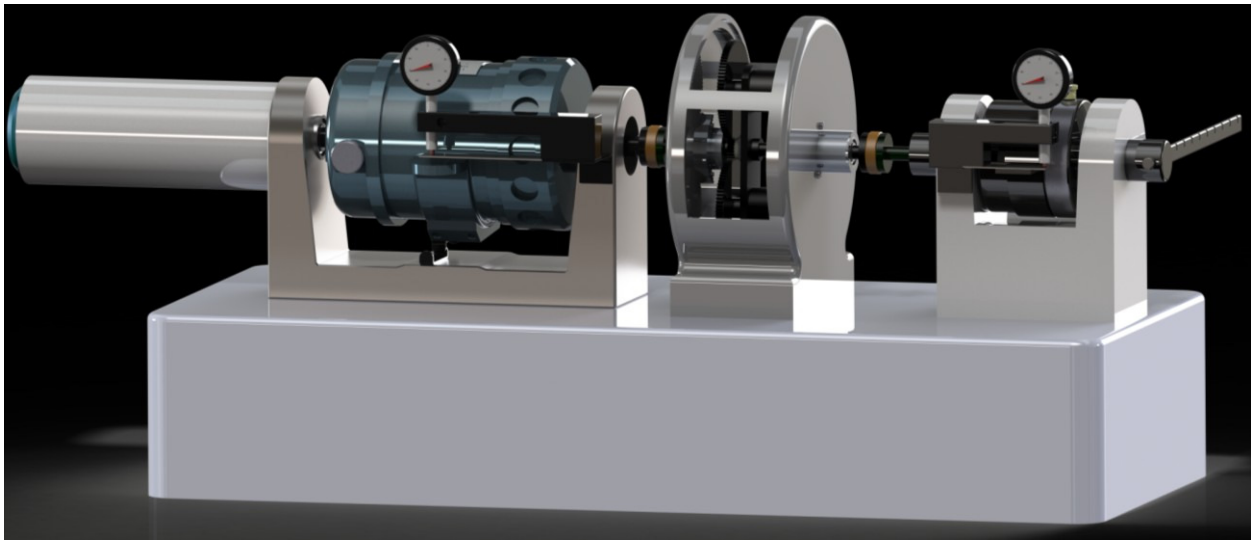

Рис. 6. Модель стенда с планетарной передачей, созданной в *SolidWorks*

Разберём подробнее эту операцию.

Сначала был создан массив нодов типа *IAnimatedMeshSceneNode*. Загрузка детали из указанной папки:

```
mesh[i] = smgr->getMesh("C:/Program Files/LW5/detali/skoba.obj");
        if (!mesh[i])  // проверка загрузки
        {
            device->drop();   //если не загрузилась, то удаляем корневой объект
            return 1;  // выходим из программы
```

```
}
node[i] = smgr->addAnimatedMeshSceneNode(mesh[i]);
if (node[i])
        {
                node[i]->setMaterialFlag(EMF_LIGHTING, true);       // свет
                node[i]->setPosition(vector3df(0,-115,-65));
                node[i]->setRotation(vector3df(90,0,0));
        }
```

С помощью функции *setPosition()* устанавливалась позиция детали, а функция *setRotation()* устанавливала ориентацию в пространстве. Для каждой модели было загружено около 60 деталей. Функция *setMaterialFlag()* отвечала за освещение. Для добавления одного источника света нужно было написать две строчки:

```
scene::ISceneNode* node12 = 0;
node12 = smgr->addLightSceneNode(0, core::vector3df(800,500,-500), video::SColorf(1.0f, 0.6f, 0.7f, 1.0f), 800.0f);
```

Было использовано три источника освещения.

Для создания интерфейса был написан класс, который отслеживал, какое событие произошло: была нажата кнопка на клавиатуре или виртуальная кнопка. Было создано 8 виртуальных кнопок: одна отвечает за выход из программы, другая за просмотр тарировочной таблицы и 6 кнопок переключают положение стационарной камеры.

Также использовалось две кнопки на клавиатуре. С помощью одной можно переключаться со стационарной камеры на камеру, управляемую с помощью стрелок и мыши. Другая даёт возможность запустить анимацию, в которой камера перемещается вокруг стенда.

Ниже представлены фотографии рабочей среды *Irrlicht* (Рис. 7а, 7б, 8а, 8б).

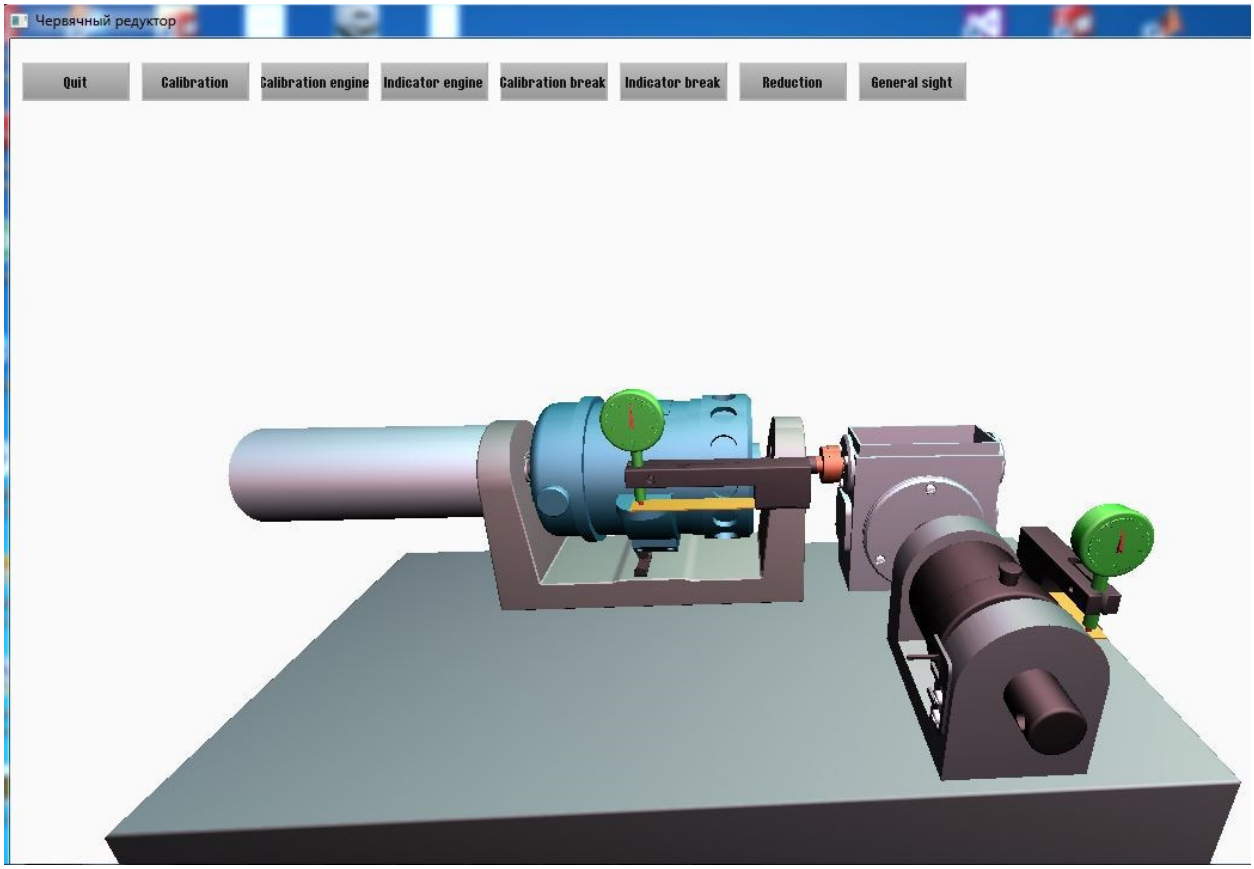

Рис. 7а. Виртуальный стенд червячной передачи в *Irrlicht*, основной вид

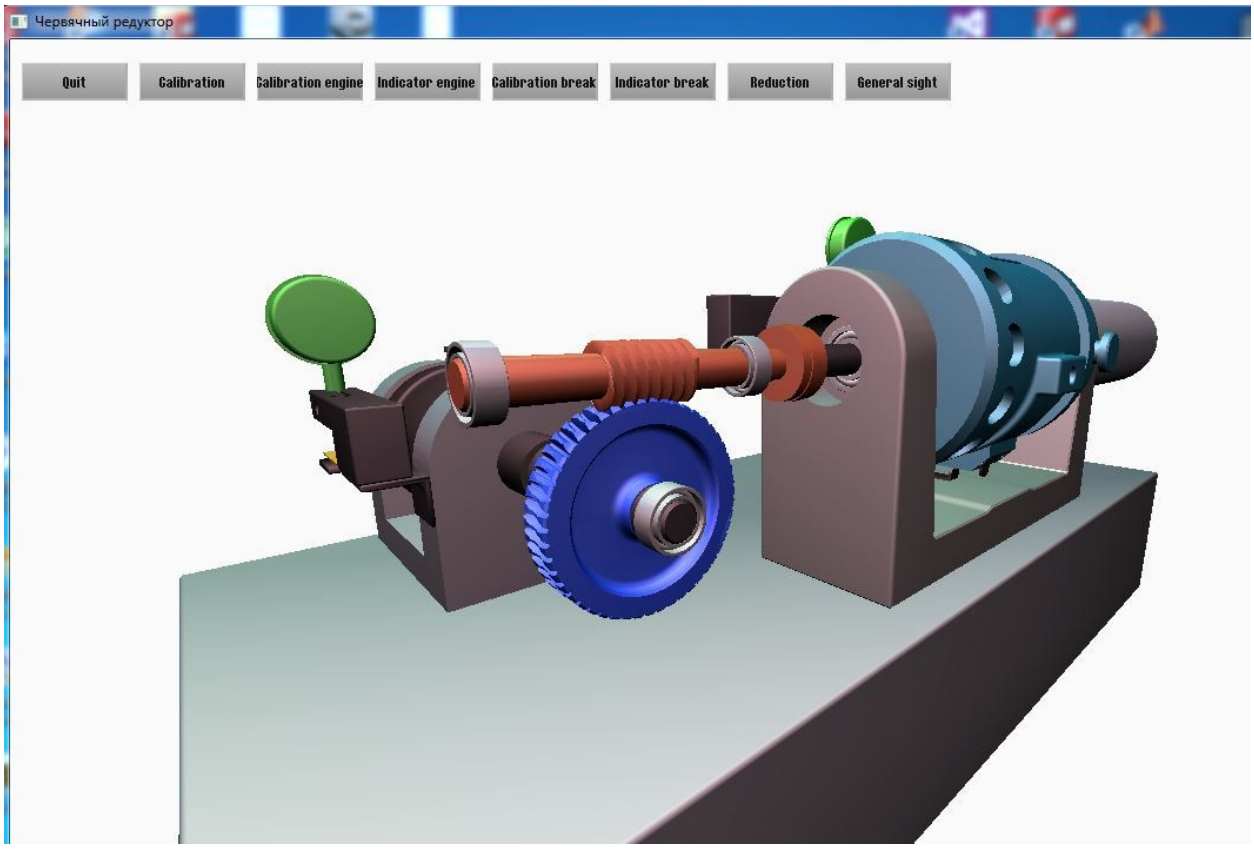

Рис. 7б. Виртуальный стенд червячной передачи в *Irrlicht*, вид сзади

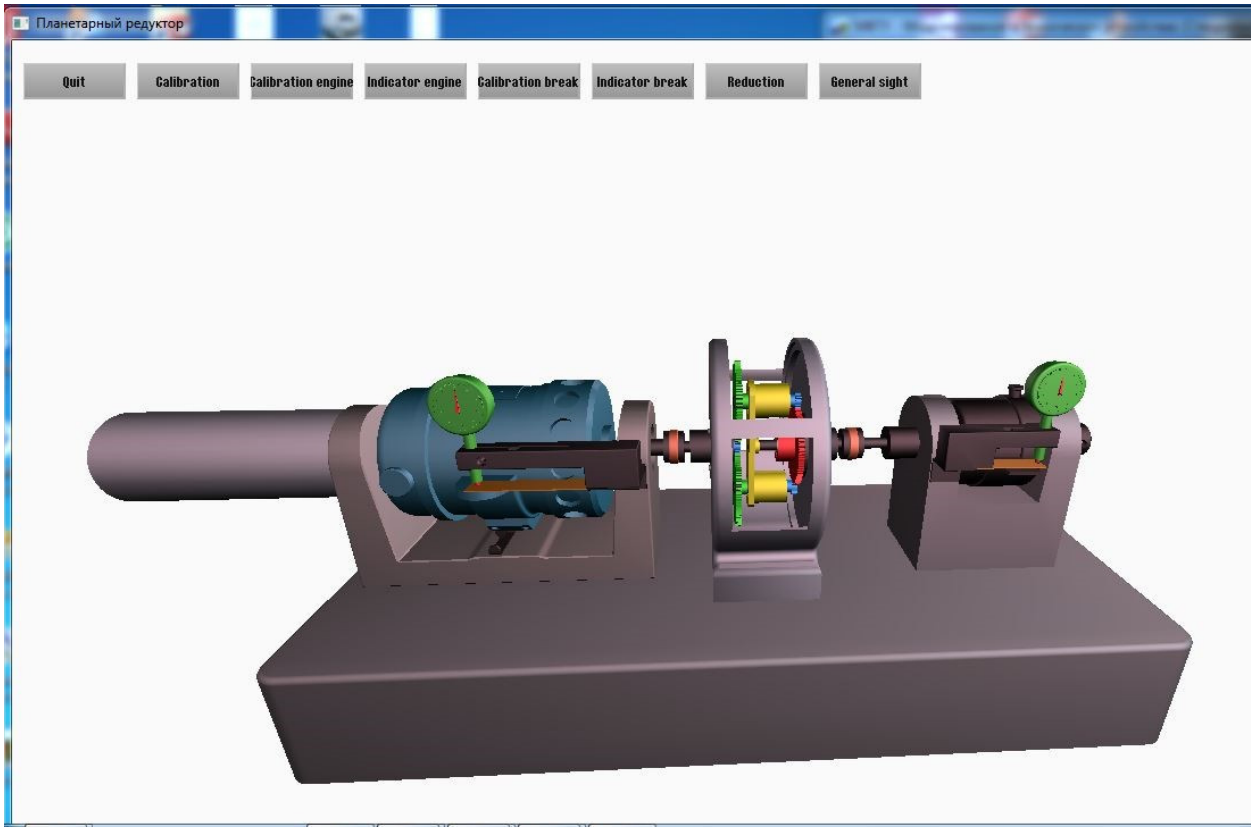

Рис. 8а. Виртуальный стенд планетарной передачи в *Irrlicht*, основной вид

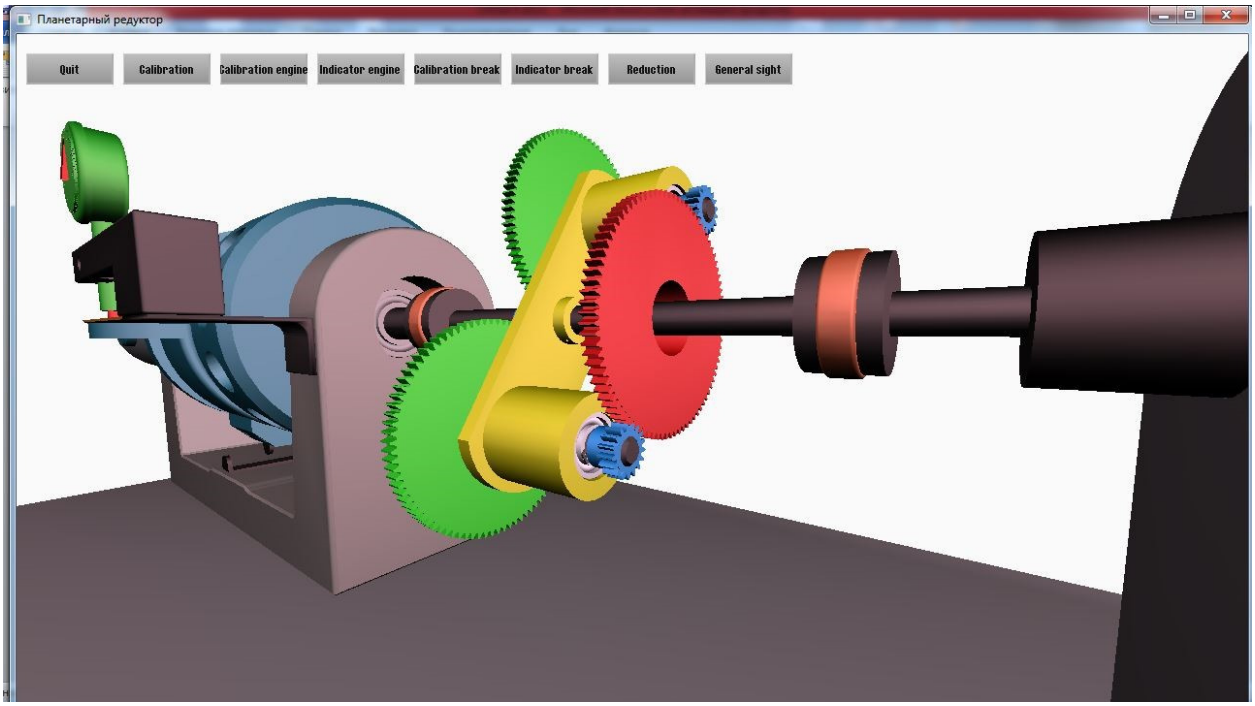

Рис. 8б. Виртуальный стенд планетарной передачи в *Irrlicht*, вид редуктора

Главный цикл программы, в котором происходит вывод на экран *while(device->run())*. В этом цикле выполнялись функции, которые позволяли получить данные из МВТУ, а именно: угол поворота выходного вала редуктора и изменение плеча двигателя и

тормоза при тарировке, а также показания индикаторов. Дальше эти данные передавались в функцию, отвечающую за анимацию на сцене, и нужные детали двигались.

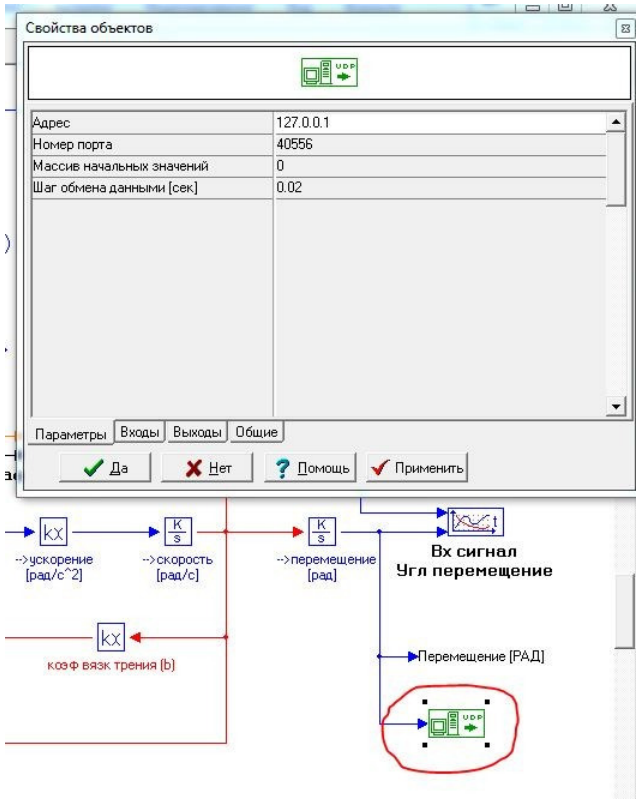

Рис. 9. Настройка *UDP* в МВТУ

Связь МВТУ и *Irrlicht*, осуществлялась с помощью протокола *UDP* и текстового файла. Угол поворота выходного вала редуктора подавался по протоколу *UDP*. *UDP* — простой, основанный на сообщениях протокол без установления соединения. Протокол такого типа не устанавливают выделенного соединения между двумя хостами. Связь достигается путем передачи информации в одном направлении от источника к получателю без проверки готовности или состояния получателя.

В окне “Свойства объектов” (Рис. 9) изменяли адрес и номер порта. Использование адреса 127.0.0.1 позволяет устанавливать соединение и передавать информацию для программ-серверов, работающих на том же компьютере, что и программа-клиент, независимо от конфигурации аппаратных сетевых средств компьютера. Таким образом, для работы клиент-серверных приложений на одном компьютере не требуется изобретать дополнительные протоколы и дописывать программные модули.

Изменение плеча двигателя и тормоза при тарировке, а также показания индикаторов, передавались через текстовый файл, так как этим данным не требуются низкие задержки при передаче данных.

Заключение.

В ходе выполнения работы разработаны *3D* модели лабораторных стендов, визуализирующих движение механических передач, тарировку датчиков, работу индикаторов. Реализованные управляемые *3D* модели лабораторных стендов не требуют значительных ресурсов от компьютера, позволяют получить хорошую детализацию и дают возможность управлять *3D* объектами с помощью математической модели. Рекомендуемые требования, предъявляемые к компьютеру:

• ОС: *Windows* ® 8 /*Windows* ® 7 / *Vista* / *Vista64*

• Процессор: *Intel Core* 2 *Duo P8400* (2,26 ГГц)

• Оперативная память: 4 ГБ

• Видеокарта: *DirectX* 9 совместимая видеокарта с памятью 512 МБ, *NVIDIA GeForce 9600M GT* или лучше

• Жёсткий диск: Не менее 200 МБ свободного места.

Список литературы

1. Остапенко Д.Г., Мурзин И.А. Моделирование лабораторной работы по курсу ОКП «Исследование КПД прямозубого зубчатого редуктора» // Молодёжный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электрон. журн. 2013. № 10. Режим доступа: http://sntbul.bmstu.ru/doc/628122.html (дата обращения 18.03.2013).
2. Гебхардт Н. Обучающие материалы по графическому движку Irrlicht Engine. Режим доступа: http://irrlicht.ru/?page_id=23 (дата обращения 20.03.2014).
3. Шилдт Г. Полный справочник по C++: пер. с англ. / под ред. С.Н. Тригуб. М.: Вильямс, 2006. 800 с. [Schildt H. C++: The Complite Reference New York : Academic Press, 2003.].