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

УДК 004.9

Тренажер для управления телами в условиях отсутствия притяжения путем симуляции движения тел в космическом пространстве

Иванов Г.Г., *студент*
Россия, 105005, г. Москва, МГТУ им Н. Э. Баумана,
кафедра «Системы обработки информации и управления»

Борзенков А.В., *студент*
Россия, 105005, г. Москва, МГТУ им Н. Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю.Е., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им Н. Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru

Вступление

В данной работе рассматривается прототип модели симулятора командного пилотирования космического аппарата в условиях низкой и нулевой гравитации, с использованием моделей боевого вооружения. В команду рассматриваемой пилотируемой единицы входят от 4-х до 6 человек.

Основной целью симуляции является отработка командного взаимодействия членов экипажа в напряженной обстановке. Для проведения таких испытаний используется модель открытого пространства в космосе. В заданной области могут находиться соответствующие препятствия в виде планет, звезд, астероидов, космического мусора. Космические объекты большой массы, такие как планеты и звезды, вносят изменения в поведение объектов, благодаря собственной высокой гравитации, что увеличивает сложность, как пилотирования космического аппарата, так и использования вооружения. Малые же объекты, в свою очередь, являются потенциальной опасностью для обшивки корабля и навесного оборудования.

Стоит отметить, что моделирование производится в рамках ближайшего будущего, что позволяет использовать теоретические модели источников энергии и материалов.

Цели и результаты

Цели.

Целью разработки данного тренажера является изучение движения инородных тел в космическом околопланетном пространстве под воздействием на них сил реактивного движения и сил притяжения со стороны астрономических объектов.

Результат.

На данный момент результатом работы являются:

- Серверное приложение (первичная версия), на котором будет происходить просчет параметров движения.
- Клиентское приложение (первичная версия), на котором будет происходить визуализация движения.
- Один из объектов, над которым будут производиться вычисления и наблюдение.

Клиент-серверная структура

Серверное приложение.

Для разработки серверной части используется язык C#. Сервером является консольное приложение, выполняющее прием, подключение и обработку клиентов.

Для передачи информации используются базовые библиотеки языка: *System.Net* и *System.Net.Sockets*.

Каждый пакет сообщения строится следующим образом: в начало сообщения помещается трехзначный код операции, затем через специальный зарезервированный символ разделения перечисляются аргументы операции.

Первичная версия протокола представлена в таблице.

Состояние\Тип	Сервер=>Клиент	Клиент=>Сервер
Сразу после подключения	000_0 = Авторизован 000_1 = Ошибка Авт-ии	000_Логин_Пароль = Запрос на авторизацию
	001 = Подтвержден переход	001_N = Переход на GameServer под номером N
Ангар	002_уровень_... = информация о уровнях классов игрока см.Приложение; 003_Idкорабля = информация о наличии корабля у игрока	002 = Клиент сообщает о готовности принять данные
	004 = Запрос принят; 005_Idкорабля_ник_ник_... = сообщение говорит о том, что очередь подошла и сразу передает информацию о кораблях и соответствующих игроках. См. Приложение	004_класс = Запрос на внесение в очередь на игру
Бой	008_id_тип_доп.инфо_Vx_Vy_Vz_Qx_Qy_Qz_Qw = Создать объект с (внимание!)Уникальным идентификатором "id", типом "тип", доп.инфо см.Приложение, Вектором положения (Vx,Vy,Vz) и поворотом, который задан кватернионом (Qx,Qy,Qz,Qw)	006 = Готов к игре, отсылается на сервер, когда клиент перешел на сцену, на которой будет разворачиваться бой.
	009_id = Уничтожить объект с идентификатором id	
	010_id_Vx_Vy_Vz_Qx_Qy_Qz_Qw = новая позиция объекта с идент-ом id	011_(0-11)_доп.инфо = Перемещение. См. Приложение
	013_ID_idтуррели_поворот = присылается если вдруг башни начали вращаться.	012_id = Клиент хочет навестись на объект id

Сервер построен на объектной модели посетителей (*Visitor*). Функциональная схема [4] представлена на рисунке 1.

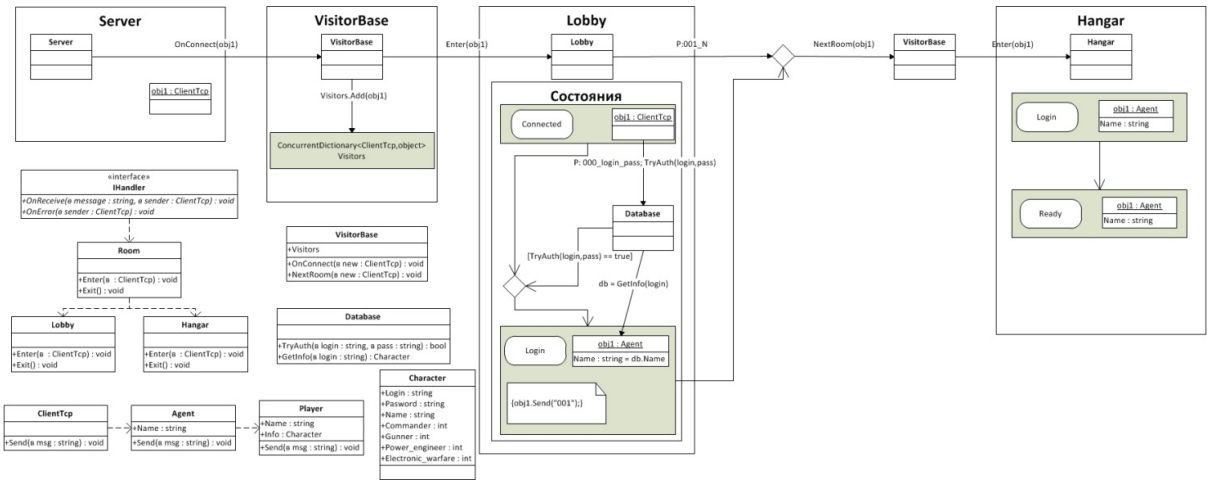

Рис. 1. Функциональная схема архитектуры сервера

Класс *Server* отвечает за начальное подключение пользователей. Содержит в себе сокет в состоянии прослушивания. После подключения пользователя, полученный сокет «оборачивается» в класс *ClientTcp* и передается в базу «посетителей» *VisitorBase*.

Класс *VisitorBase* отвечает за хранение и перенаправление посетителей. Содержит в себе потокобезопасную коллекцию для хранения экземпляров *ClientTcp* и набор функций для перемещения их между этапами.

Этапами называются классы, наследованные от абстрактного класса *Room*. Каждый такой класс поддерживает объектно-ориентированную модель сервера, разделяя пространство симулятора на несколько этапов (зон). Также, он является целью обратной связи экземпляров класса *ClientTcp* (или классов, наследованных от него). Поэтому, содержит набор функций, реализующих отведенную ему механику тренажера.

Класс *ClientTcp* по пути между различными зонами может быть преобразован в иные, унаследованные от него классы. Так, поступая в класс *Lobby*, отвечающий за авторизацию и распределение по кластерам сервера, *ClientTcp*, после прохождения пользователем авторизации, преобразуется в класс *Agent*, содержащий в себе информацию о пользователе и его персонаже, загруженную из базы данных.

Классы *Lobby*, *Hangar* и *Scene* являются тремя основными этапами, в которых пользователи взаимодействуют друг с другом и окружением. Классы *Hangar* и *Lobby* являются статическими классами. В отличие от них, класс *Scene* может обладать множеством экземпляров. Любая группа пользователей может запросить у сервера персонально выделенную зону для проведения симуляции, которой и являются экземпляры Scene.

Класс *Hangar* (ангар) представляет пользователю возможность выбрать желаемый космический аппарат, участвовать в обсуждениях с другими пользователями (чат) и перейти на следующий этап *Scene* – этап симуляции.

Класс *Scene* содержит в себе набор функций, которые реализуют симуляцию взаимодействия объектов в космическом пространстве на основании физических законов. При начале симуляции создается виртуальное пространство, состоящее из экземпляров классов, унаследованных от класса *SObject* (базовый физический элемент для взаимодействия объектов в виртуальном пространстве).

Клиентское приложение.

Клиентская часть разработана в среде Unity с использованием скриптов, написанных на языке C#. Unity это инструмент для разработки двух- и трёхмерных приложений, работающий под операционными системами Windows и OS X. Приложения, созданные с помощью Unity (www.unity3d.com), поддерживают DirectX и OpenGL.

Конкретно этот игровой движок был выбран по нескольким причинам. Во-первых, это свободно распространяемый программный продукт. Во-вторых, редактор имеет простой и свободно настраиваемый интерфейс, не создающий проблемы при работе с ним.

Обзор используемого ПО

Краткий обзор программы.

1) Рассмотрим интерфейс на рисунке 2.

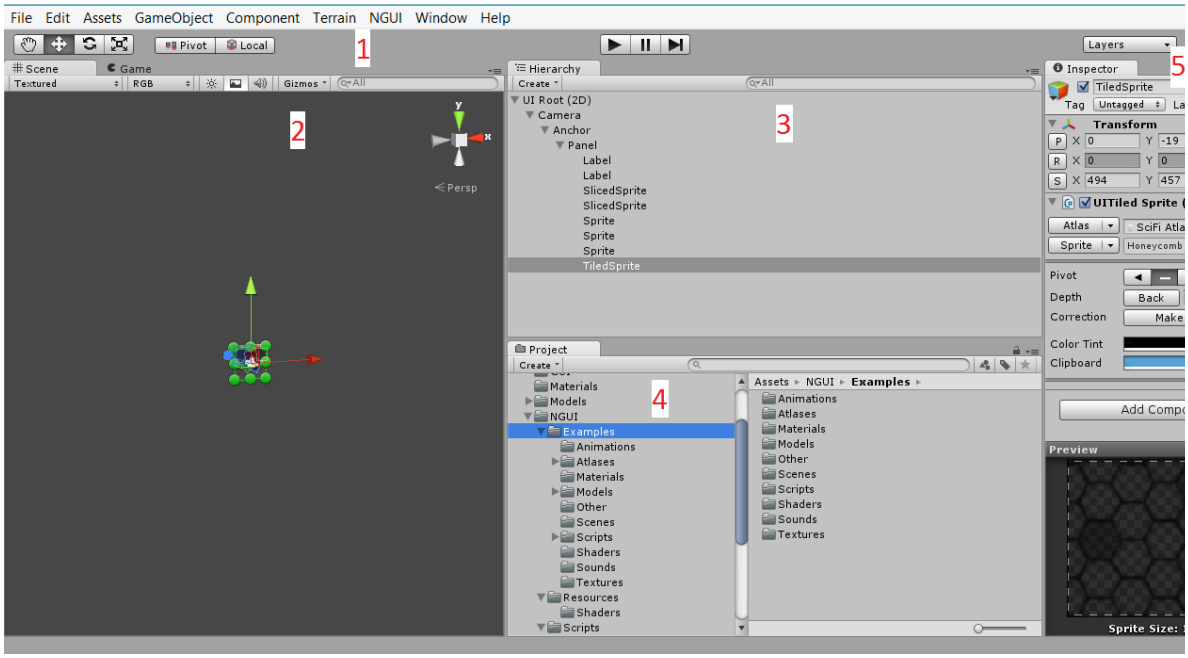

Рис. 2. Интерфейс программного продукта Unity

1. Панель инструментов. Содержит пять основных компонентов.
   1.1. Инструменты трансформации – используются в окне сцены.
   1.2. Переключатели окна гизмо – влияют на содержимое окна сцены
   1.3. Кнопки *Play*/*Pause*/*Step* – используются в окне проигрывателя.
   1.4. Выпадающее меню [[*Layers*|*Layers*]] – отвечает за то, какие объекты должны быть отображены в окне сцены
   1.5. Выпадающее меню *Layout* – контролирует расположение всех окон
2. Окно сцены и проигрывателя.
   2.1. Окно сцены используется для выделения и позиционирования всех элементов проекта. Панель управления сценой нужно изменения различных настроек.
   2.2. Окно проигрывателя отрисовывается из камеры (камер) в вашей сцене. Это отображение финального, опубликованного приложения. На панели управления проигрывателем можно управлять видимыми характеристиками окна (графикой, производительностью и т.п.) а также видимостью объектов.
3. Иерархия. Это окно содержит все объекты в текущей сцене. Существует возможность выбрать объекты в иерархии, и перетащить один объект на другой, для создания дерева связей.
4. Обозреватель проекта. В левой панели обозревателя находится структура папок проекта в виде иерархического списка. При выборе папки в этом списке в панели справа отобразится её содержимое.
5. Инспектор. В этом окне отображается детальная информация о текущем выбранном объекте, включая все его компоненты и их свойства. Существует возможность изменять функционал объектов в сцене, не прибегая к изменению самих объектов (компонентов), например, можно изменить переменные скрипта, без изменения самого скрипта.
6. Другие окна (консоль, окно анимации, и т. п.)

2) Принципы работы в движке Unity.
   1. Объекты. Объекты приложения, размещенные на сцене, представляют собой контейнер для различных составляющих. Чтобы понять, что такое игровой объект в данном контексте, необходимо понять, какие компоненты он может включать.

1.1. Любой созданный объект создается с уже связанным компонентом – Transform. Этот компонент характеризует объект и определяет его положение вращение и масштаб в симуляционном пространстве и в окне сцены. Также этот объект вводит важный принцип Unity - наследование. То есть к свойствам объекта, таким как, положение, масштаб и вращение, обращаются через компоненту *Transform* путем операции точки (*.Position, .Rotation, .Scale*).

1.2. Существует огромное количество других компонентов, например, объект «камера» создается с уже связанными компонентами – *Camera* (отображает пользователю симулированный мир), *GUILayer* (отвечает за отрисовку двухмерного пользовательского интерфейса), *Flare Layer* (отвечает за блики, отображаемые на камере), и *Audio Listener* (получает данные с входящего устройства звука и проигрывает звуки через динамики).

2. Сцены. Сцены содержат объекты. Они могут использоваться для создания меню с интерфейсом, разных последовательностей симулятора (авторизация, подключения, обсуждения, тренажер). Можно считать каждый файл сцены отдельным уровнем. В каждой сцене можно разместить объекты создаваемого тренажера, как имеющие реальное воздействие на окружающие объекты, так и декоративные.

2.1. Объекты можно создавать с помощью встроенных инструментов, также возможен импорт или использование префабов (особый тип данных, хранящих объект со всеми компонентами и значениями свойств, используется в качестве шаблона для создания экземпляров)

2.2. Когда объект создан на рабочей сцене, к нему нужно добавлять компоненты (так, как было описано ниже) в том числе и скрипты (см. ниже).

2.3. После создания объекта возможно изменения его расположения (см. компонент *Transform*)

2.4. В качестве объектов можно добавлять камеры, через которые будут происходить наблюдение за симулированным пространством.

2.5. Также в сцену можно добавлять источники света, что добавляют симулированному пространству реалистичности.

3. Сприпты. Одним из компонентов, которые можно добавлять к объекту, является скрипт. Скрипт – это описание действий, написанных на высокоуровневом языке программирования, которые выполняются системой.

Рассмотрим скрипт на примере скрипта, наложенного на объекты в симулированном мире и выполняющего функции наведения объектов на цели.

3.1. Экземпляр скрипта присоединен к пустому объекту, который является хранилищем скриптов в данной сцене.

3.2. В самом скрипте происходит поиск тех объектов, которые должны двигаться (вращаться) и объектов (целей), на которые может происходить наведение с помощью тегов (уникальных свойств, накладываемых на объект).

3.3. Навигация происходит с помощью графического пользовательского интерфейса, такого, как на рисунке 3.

Рис. 3. Пример пользовательского интерфейса

3.4. Вращение в каждый момент времени задается кватернионом (системой комплексных чисел, образующих векторное пространство размерностью четыре)

3.5. Кватернион определяется начальной ориентацией объекта и вектором поворота (в Unity класс *Vector3* описывает трехмерный вектор).

3.6. Вектор поворота задается как разница между двумя векторами – вектором направленным по центру поворачиваемого объекта (*Gun.transform.forward*) и вектором соединяющим центры двух объектов (поворачиваемого и направляющего, *Vector3* vec2 = *Target.transform.position* - *Gun.transform.position*).

3.7. Скриншот работы скрипта на рисунке 4.

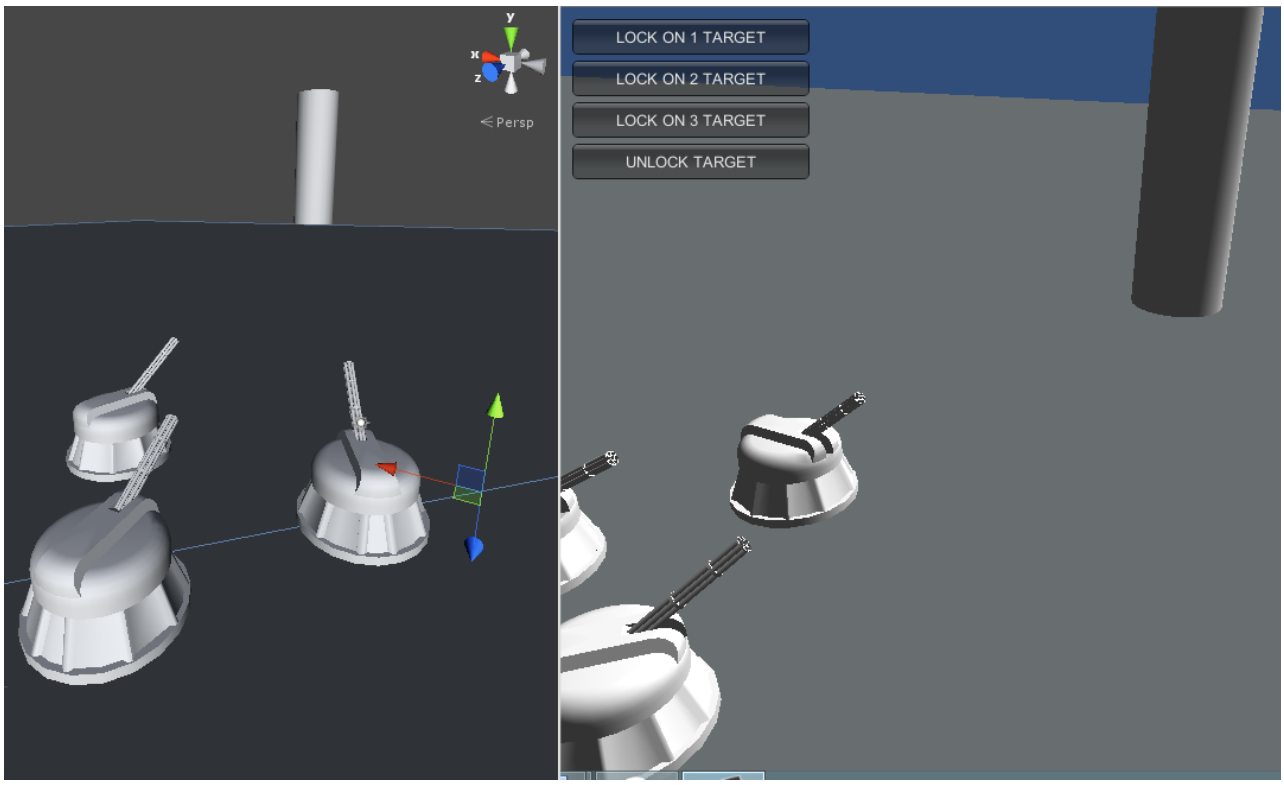

Рис. 4. Пример работы скрипта, ориентирующего одну группу объектов на другой

3) Взаимодействие клиента с сервером на текущий момент выглядит следующим образом в нескольких сценах.

1. Сцена авторизации (*Lobby*) показана на рисунке 5.

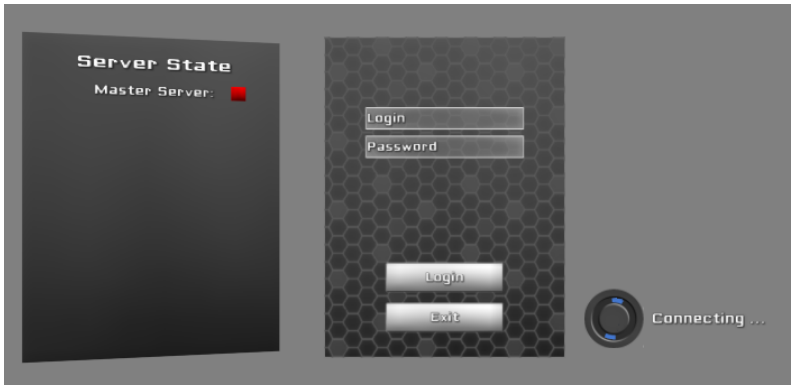

Рис. 5. Сцена авторизации

2. Сцена выбора корабля и команды (*Hangar*) на рисунке 6.

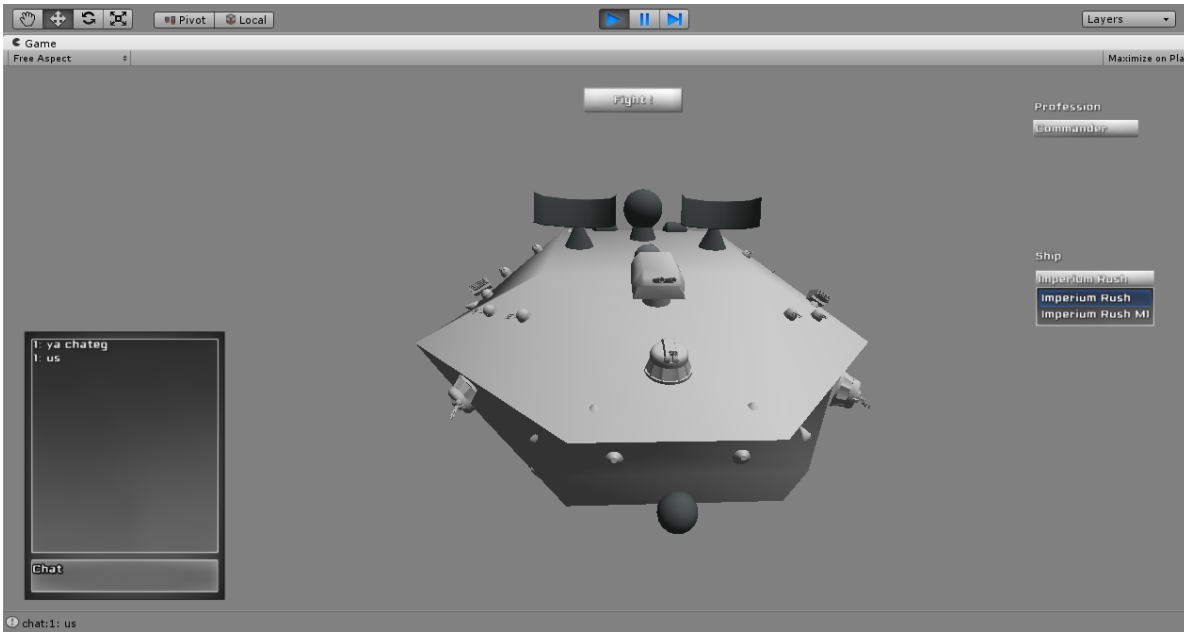

Рис. 6. Сцена выбора корабля и команды

3. Сцена симуляции (*Scene*) на рисунке 7.

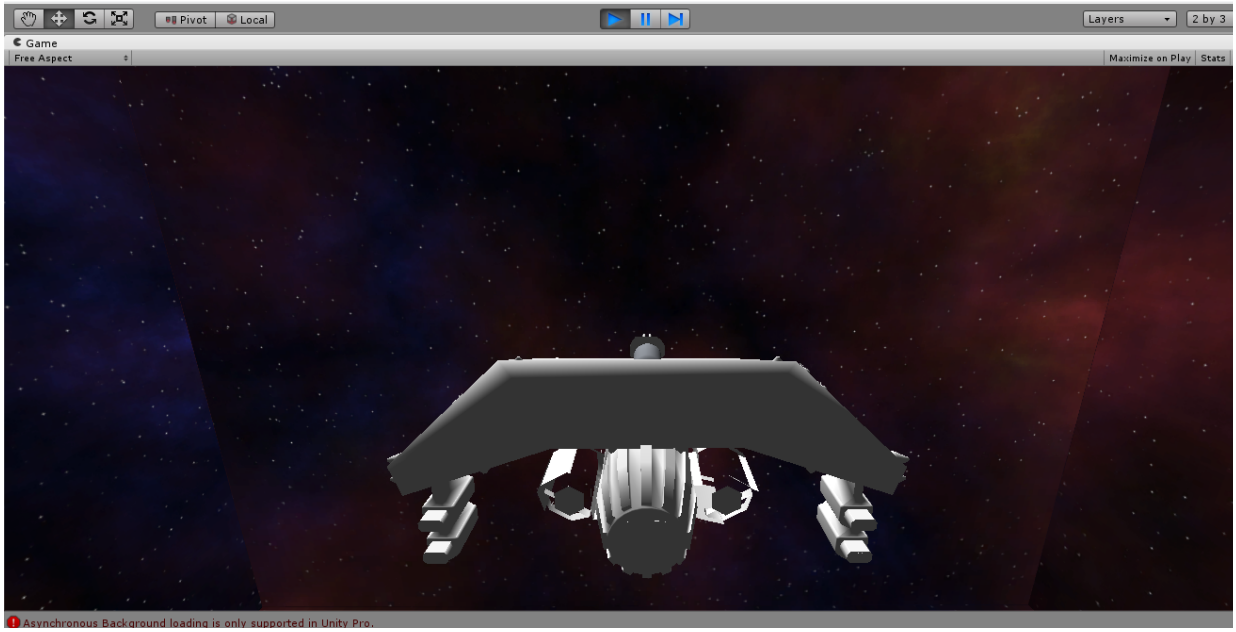

Рис. 7. Сцена симуляции

Создание объектов.

Тела, представленные на тренажере, разработаны самостоятельно с помощью программного пакета Cinema 4D фирмы Maxon (www.maxon.net) для создания трехмерной графики и анимации. Cinema 4D является универсальной комплексной программой для создания и редактирования трёхмерных эффектов и объектов. Поддерживает анимацию и

высококачественный рендеринг. Отличается более простым интерфейсом, чем у аналогов, и встроенной поддержкой русского языка.

После изучения этой программы появляется необходимая информацию для направления 3D. Приобретаются навыки владения основными элементами программы, такими как создание простейших объектов, их модификацию и связывание различными способами. Рассмотрим подробнее программу Cinema 4D.

1) Рассмотрим основные панели программы на рисунке 8.

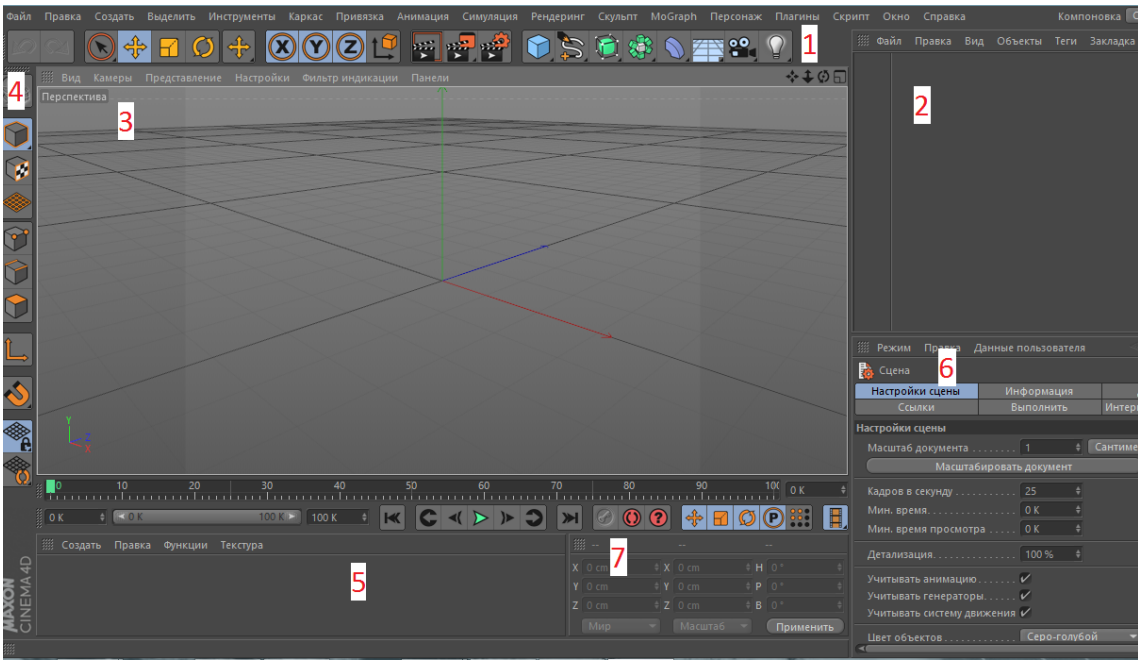

Рис. 8. Окно программы Cinema 4D

1. Командная группа. На этой панели можно выбирать параметры курсора, изменять и фиксировать рабочие оси системы координат, работать с рендером, создавать простейшие объекты, добавлять эффекты, плоскости, камеры, свет.
2. Менеджер объектов. Здесь будут появляться все объекты, которые есть на сцене, а также наложенные эффекты, например текстура. Эти объекты будут появляться в виде дерева, где к ним можно будет применять различные функции. На этой же вкладке можно произвести импорт уже существующих объектов/текстур.
3. Сцена. Тут находится тот вид, над которым работает пользователь. Он позволяет изменять объекты с помощью функций программы путем воздействия курсора мыши, который имеет различное применение. На этом виде можно изменять настройки камеры, добавлять еще камеры в настраиваемой комбинации для удобной работы.

4. Командная панель. Здесь находятся функции, которые влияют на все объекты, находящиеся на сцене.
5. Менеджер материалов. С помощью этой вкладки можно создавать и удалять материалы и текстуры, а также накладывать их на объекты путем переноса их на объект в менеджере или на сцене.
6. Менеджер атрибутов, включая инструмент. Здесь можно производить множество действий по изменению свойств различных объектов, функций и действий.
7. Свойства объекта. Интерактивная панель для получения и изменения данных об объекте, таких как размер, вращение и позиция.

2) Рассмотрим простейшие объекты и способы их изменения.

1. Создание объекта куб. На панели командной группы выберем значок «Создать объект» и из предложенного списка выберем «Объект Куб». В центре сцены появится куб.
2. Конвертация объекта. Объект (куб) на начальном этапе создания является параметрическим объектом. Чтобы начать моделирование, необходимо произвести преобразование объекта в полигональный с помощью соответствующей кнопки на командной панели.
3. Добавление подобъектов. Теперь доступна возможность изменения или перемещения отдельных точек и поверхностей для конвертированного объекта. Для этого мы можем использовать объект «Разбивка поверхности», создав его нажатием на кнопку «Создать объект» -> «Разбивка поверхности» на панели командной группы.
4. Связка объектов. Добавим куб в качестве подобъекта к объекту «Разбивка поверхностей» в менеджере объектов путем перетаскивания мышкой в том же менеджере.
5. Результат. В результате объект получает плавные формы внешней геометрии без изменения действительной разбивки. Этот эффект будет сохраняться при изменения частей куба по отдельности, например, вытягивании точки. См. рис. 9.

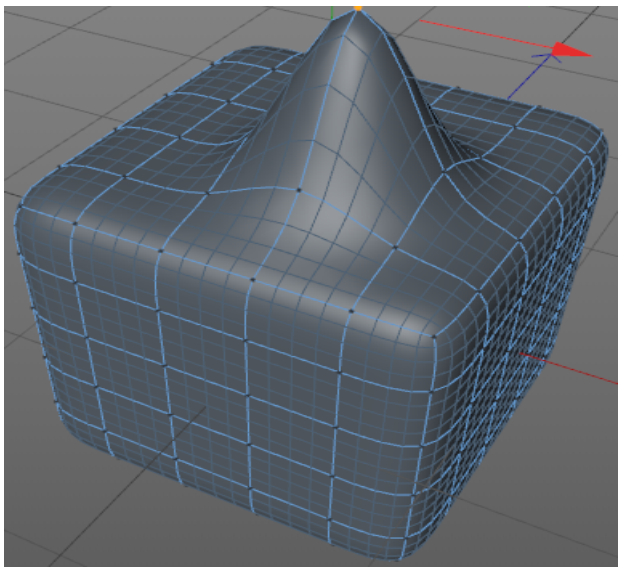

Рис. 9. Изменение объекта

3) Рассмотрим поэтапное создание объекта «Пушечная башня» на рис. 10.

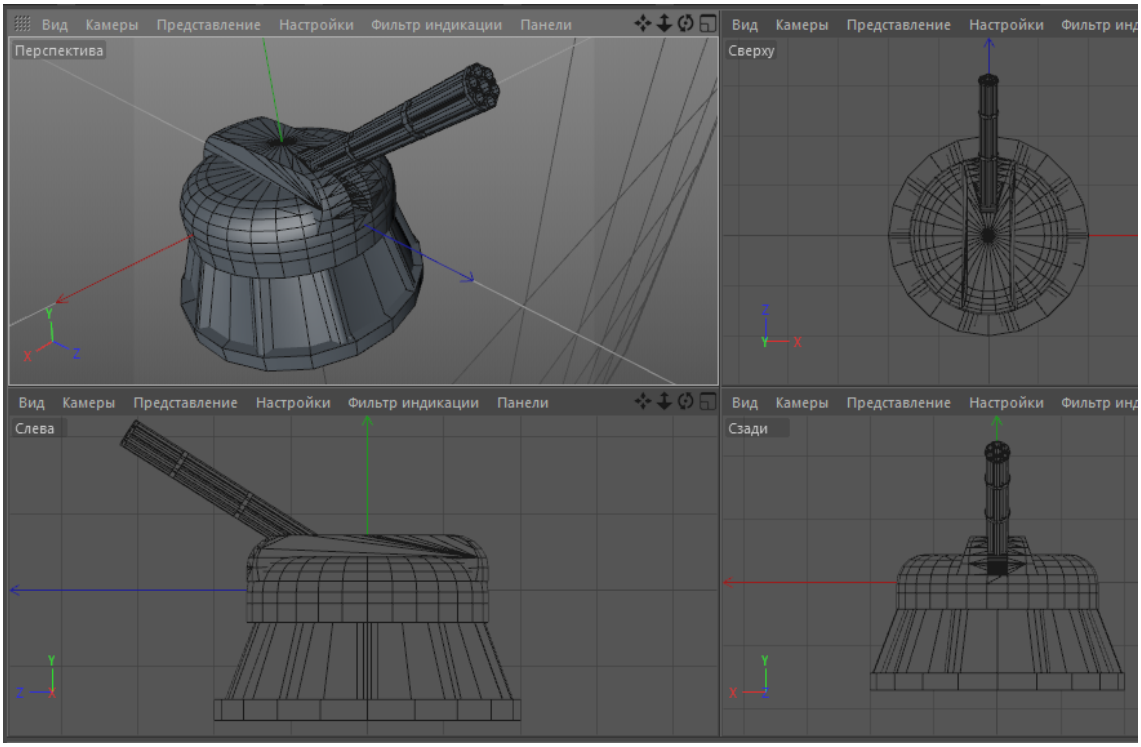

Рис. 10. Пушечная башня

1. Создание платформы.

1.1.Создание основания.

1.1.1. Создание цилиндра.

1.1.2. Создание конуса.

1.1.3. Разрезание конуса на полигоны с помощью функции «Нож».

1.1.4. Выдавливание разрезанных полос.

1.1.5. Соединение цилиндра и конуса. См. рис. 11.

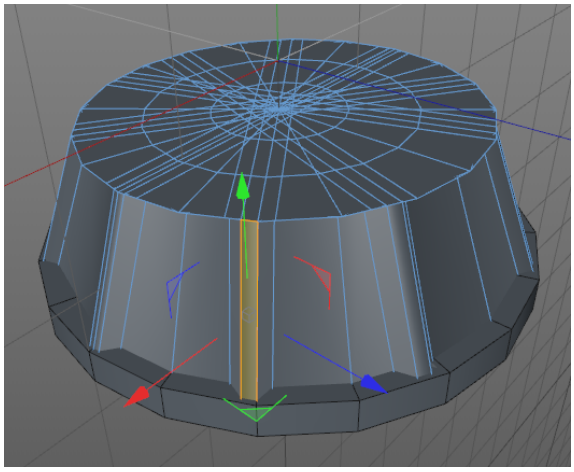

Рис. 11. Пример соединения объектов

1.1.6.

1.2.Создание башни.

1.2.1. Создание цилиндра с фаской сверху.

1.2.2. Создание фрагмента трубы.

1.2.3. Применение операции «Булев» к двум вышеописанным объектам. См. рис. 12.

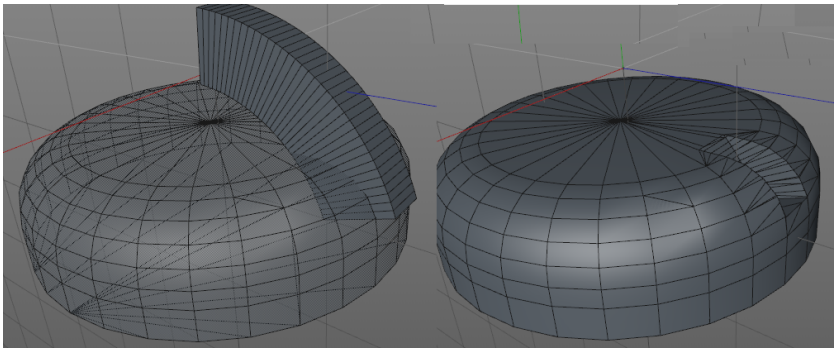

Рис. 12. Операция «Булев»

2. Создание орудия.

2.1.Создание платформы.

2.1.1. Создание цилиндра с фаской с двух сторон.

2.1.2. Создание четырехугольного параллелепипеда на основе куба.

2.1.3. Применение операции «Булев» к двум вышеописанным объектам

2.1.4. Создание фрагмента трубы.

2.1.5. Применение операции «Булев» к двум вышеописанным объектам (2.1.3, 2.1.4) См. рис. 13.

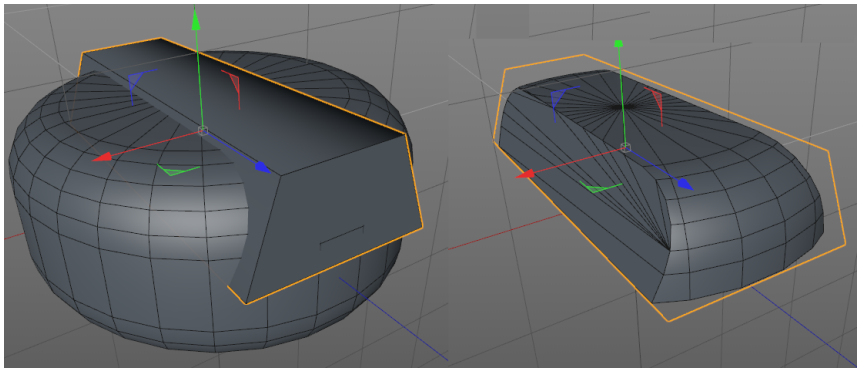

Рис. 13. Операция «Булев»

2.2.Создание стволов.

2.2.1. Создание трубы.

2.2.2. Выполнение операции «Разместить» над этой трубой.

2.2.3. Создание четырех цилиндров, выполняющих вид утолщения.

2.2.4. Выполнение операции «Булев» над предыдущими двумя объектами.

4) Модель космического корабля, построенная в Cinema 4D (без пошагового описания). См. рис. 14.

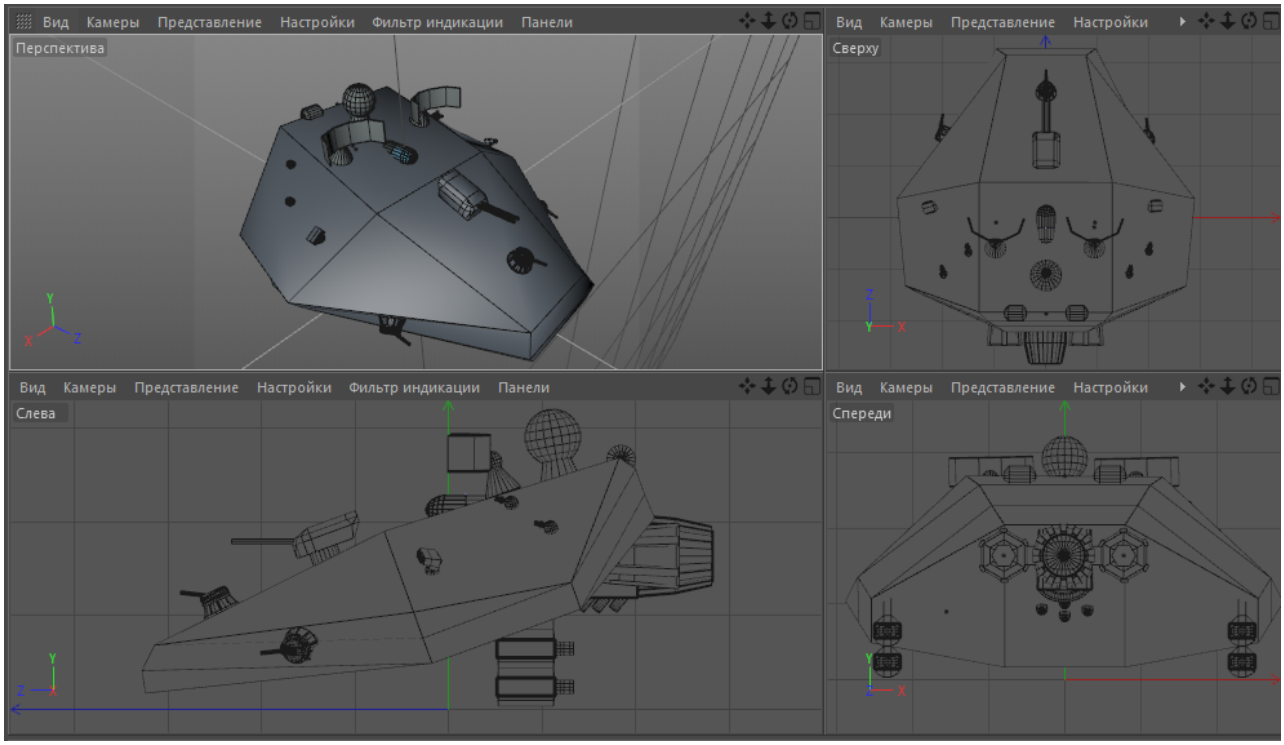

Рис. 14. Модель космического корабля.

Заключение

В заключении, хотелось бы отметить, что целью, преследуемой командой разработчиков при создании проекта, является построение высоко реалистичной модели поведения космических аппаратов в условиях изменяющейся гравитации для улучшения навыков групп лиц при ориентации в трехмерном пространстве, взаимодействии внутри команды как одной пилотируемой единицы, так и нескольких.

Список литературы

1. Microsoft Developer Network. Режим доступа: https://msdn.microsoft.com/ (дата обращения 26.02.2015).
2. Официальный сайт Unity3D. Режим доступа: http://unity3d.com/ (дата обращения 26.02.2015).
3. VideoSmile - всё о визуальных эффектах и подвижной графике. Режим доступа http://videosmile.ru/ (дата обращения 26.02.2015).
4. Самохвалов Э.Н., Ревунков Г.И., Гапанюк Ю.Е. Использование метаграфов для описания семантики и прагматики информационных систем. Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение». 2015. Выпуск № 1. Режим доступа: http://vestnikprib.bmstu.ru/catalog/icec/infth/673.html.