УДК 004.415.25 **Технические науки**

Ильичев Владимир Юрьевич, *к.т.н., доцент кафедр «Тепловые двигатели и гидромашины» и «Мехатроника и робототехнические системы»*
Калужский филиал ФГОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), г. Калуга, Россия, научный руководитель
Жариков Артем Андреевич, *студент кафедры «Тепловые двигатели и гидромашины»*
Калужский филиал ФГОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), г. Калуга, Россия, соавтор

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ОБЪЕКТОВ В UCHRONIA PROJECT BLENDER GAME ENGINE (UPBGE)

Аннотация: В работе рассматривается методика создания перемещаемых 3D объектов в движке Blender Game Engine (BGE) новейшей сборки, называемой Uchronia Project (UPBGE). Кроме симуляции движения рассматривается настройка параметров освещения объектов, положения фиксирующей камеры и прочих свойств моделируемой сцены. Главным же разработанным этапом настройки движка UPBGE является работа с логическим редактором выполняемых объектом действий. Рассмотренная в статье методика может являться основой создания наглядных игровых и обучающих приложений.

Ключевые слова: симуляция движения, 3D моделирование, игровая сцена, движок UPBGE, редактор логических действий.

Annotation: The paper discusses how to create moving 3D objects in the Blender Game Engine (BGE) of the latest assembly, called the Uchronia Project

(UPBGE). In addition to motion simulation, we consider setting the lighting parameters of objects, the position of the fixing camera and other properties of the simulated scene. The main developed step of configuring the UPBGE engine is to work with the logical editor of the actions performed by the object. The methodology discussed in the article can be the basis for creating visual gaming and training applications.

Keywords: motion simulation, 3D modeling, game scene, UPBGE engine, logical action editor.

Введение

В современной реальности всё более популярными становятся системы 3D моделирования объектов [1], причём с помощью компьютера имитируются не только статичные, но и динамически изменяющиеся сцены. Такие сцены называют визуальной динамической моделью движения одного или нескольких объектов.

Особенностью современных систем моделирования динамических сцен является имитация не только движения объекта [2], но и его освещения (часто с использованием нескольких различных источников света) и соответственно отбрасываемых объектом теней, положения основания, по которому перемещается объект, а также камеры (или камер), фиксирующих движение. При этом каждая упомянутая симуляция обычно обладает огромным количеством настроек (в частности, можно выбрать тип камеры, её фокусное расстояние, коэффициент приближения, положение относительно связанной с объектом и глобальной системы координат и другие свойства).

Область применения динамических симуляций движения не ограничивается играми или анимацией [3]; они всё чаще используются в образовательном процессе для большей наглядности манипуляций над объектом (самые известные динамические симуляторы используются при обучении управления военной техникой, автомобилем и прочим транспортным оборудованием). Перспективность и необходимость такого подхода уже давно

ни у кого не вызывает сомнений. Однако проблемой является доступность выполнения динамического моделирования для широкого круга заинтересованных лиц, т.к. только недавно появились бесплатные и достаточно простые для освоения средства создания подвижных и управляемых моделей. К одной из наиболее доступных систем комплексного моделирования практически любых систем на большом количестве системных платформ относится программа Blender 3D, которая уже рассматривалась авторами [4], однако только в применении к созданию статических систем, либо динамических, но уже встроенных в качестве тренажеров в саму программу.

Целью данной статьи является изложение разработанной авторами методики создания логически управляемых динамических моделей с помощью специальной (но также бесплатной) версии программы Blender 3D, включающей в себя специальный движок для симуляции движения при выполнении пользователем определённых действий. Эта версия называется UPBGE (Uchronia Project Blender Game Engine) и была создана сравнительно недавно – в конце 2015 г [5]. С тех пор она была значительно усовершенствована по удобству использования, по быстродействию. а также в неё постоянно добавляются новые функции, согласно потребностям разработчикам современных приложений.

Симулятор UPBGE позволяет также выполнять действия по редактированию объектов, в нём можно использовать стороннюю векторную или растровую графику, текстуры и многое другое. Создаваемый промежуточный проект можно сохранять в виде файла Blender, а конечный проект (игру или симулятор) – в формате исполняемого файла.

Материал и методы исследования

Авторами был разработан следующий порядок создания и запуска любого проекта UPBGE, состоящий из нескольких последовательных стадий:

1. Отрисовка статичной сцены, включающей в себя систему координат, основание для размещения объектов, сами объекты (обычно являющиеся векторными, трёхмерными и твердотельными), определение расположения и

типа источников освещения, положения камеры (или нескольких камер) для рендеринга изображения [6].

2. Создание программы работы с объектами в специальном редакторе логики путём определения связей между тремя типами блоков: сенсоров, контроллеров и актуаторов [7]. Сенсоры предназначены для физического принятия команд пользователя, контроллеры – для выполнения того или иного логического действия, а актуаторы – для связи одиночного логического действия с определённым поведением объекта. Чаще всего в качестве сенсоров используются клавиши компьютерной клавиатуры, датчики или клавиши мыши. Контроллер – связанная с воздействием на тот или иной сенсор логическая команда или даже выполнение целой компьютерной программы, например, на языке Python [8]. В качестве актуатора могут выступать самые разнообразные действия – команды изменения формы или расположения объекта, подача звукового сигнала или текстового сообщения, удаление объекта и многие другие. При выборе определённого актуатора чаще всего во всплывающем окне также необходимо задать ряд характеризующих его параметров (например, в случае перемещения объекта выбирается дискретное приращение координат его положения и углов поворота относительно системы координат).

3. Создание последовательных связей между логическими блоками (сенсорами, контроллерами и актуаторами). При этом один контроллер можно связать с несколькими сенсорами и актуаторами.

4. В случае необходимости компиляция созданной на предыдущих шагах программы в виде исполняемого файла.

5. Задание параметров запуска проекта (отдельно для встроенного и оконного плееров): графическое разрешение отображаемого и записываемого в файл динамического изображения, частота обновления его кадров, качество воспроизводимых звуков, выбор источников освещения, наличия или отсутствия отбрасываемых объектами теней, степень увеличения изображения выбранной камерой и т.д.

6. Собственно запуск проекта нажатием кнопки старт в случае отработки его в программе UPBGE (без компилирования в исполняемый файл exe), совершения действий с помощью заданных сенсоров и наблюдение за поведением объектов на экране (возможно, с одновременной записью в видеофайл).

Пример расчёта

В качестве примера рассмотрим процедуру создания с помощью UPBGE простого симулятора движения [9], состоящей в произвольном перемещении объекта на горизонтальной плоскости. Каждый этап работы над создаваемой динамической сценой проиллюстрируем скриншотами.

Как было определено последовательностью созданной и описанной выше методики, вначале необходимо создать статическую сцену симулятора. Она изображена на рис. 1.

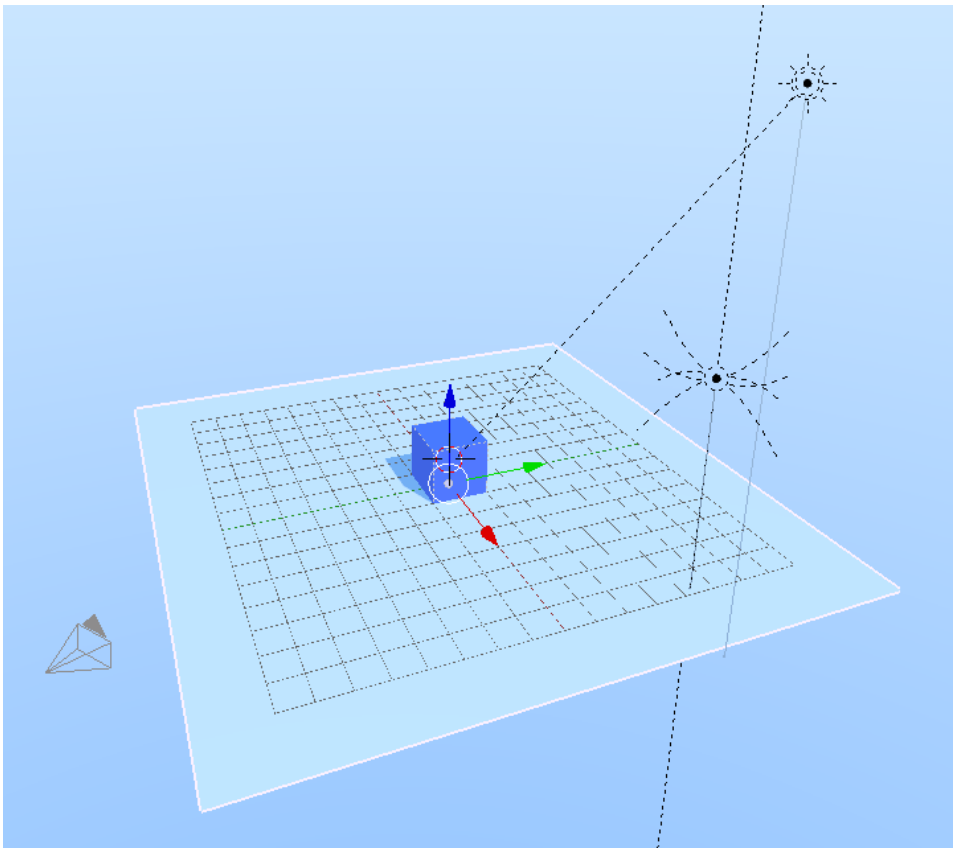

Рис. 1. Первоначальная статическая сцена для разрабатываемой игры.

На рисунке можно выделить следующие элементы сцены: оси глобальной

системы координат, опорная горизонтальная плоскость, три источника света (условно показан их тип и направление освещения), сам объект в виде куба, в левой области сцены видна камера.

Примем, что создаваемым в дальнейшем главным направлением движения куба будет являться движение вдоль оси координат *y*, обозначенной зелёной стрелкой. Так как куб – симметричная фигура, форма которой определена координатами расположения вершин – узлов сеточного 3D представления, с помощью встроенных в UPBGE средств создания и редактирования сеток изменим форму данного объекта в направлении оси *y* для отличия данного направления от всех остальных. Результат преобразования куба представлен на рис. 2.

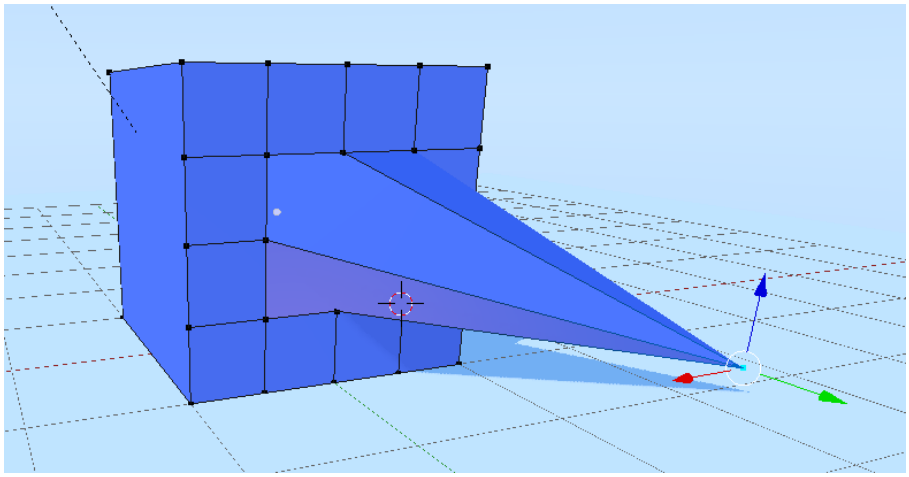

Рис. 2. Результат преобразования сеточной модели объекта - куба.

Первоначально куб был представлен 8 вершинами-узлами. Для редактирования формы этого объекта, была выделена его грань, расположенная по направлению оси *y*, и дважды разбита на более мелкие ячейки сетки. Таким образом, при первом разбиении из одной квадратной области были созданы 4, а при втором разбиении 16 ячеек. Затем были выделены 4 центральные области и путём вытягивания по направлению оси *y* создана своеобразная объёмная «стрелка». Необходимо отметить, что программа UPBGE содержит также множество прочих средств и способов редактирования объектов, и здесь приведён лишь один из них.

После этого была создана простейшая логическая цепочка «сенсор-контроллер-актуатор», позволяющая объекту перемещаться вперёд на небольшое расстояние (относительное) y=0.1 при нажатии клавиши клавиатуры «W» и запущен процесс симуляции созданной модели. Однако, при этом оказалось, что при длительном нажатии клавиши объект быстро исчезает из поля зрения камеры по причине, что камера остаётся статичной (жёстко привязана к определённым глобальным координатам, не связанным с объектом). Поэтому на следующем этапе система координат камеры была связана с координатами объекта, размещена ближе к нему, а поле её зрения развёрнуто по направлению оси y. На рис. 3 показан вид сверху, отображающий созданное взаимное расположение камеры и управляемого объекта.

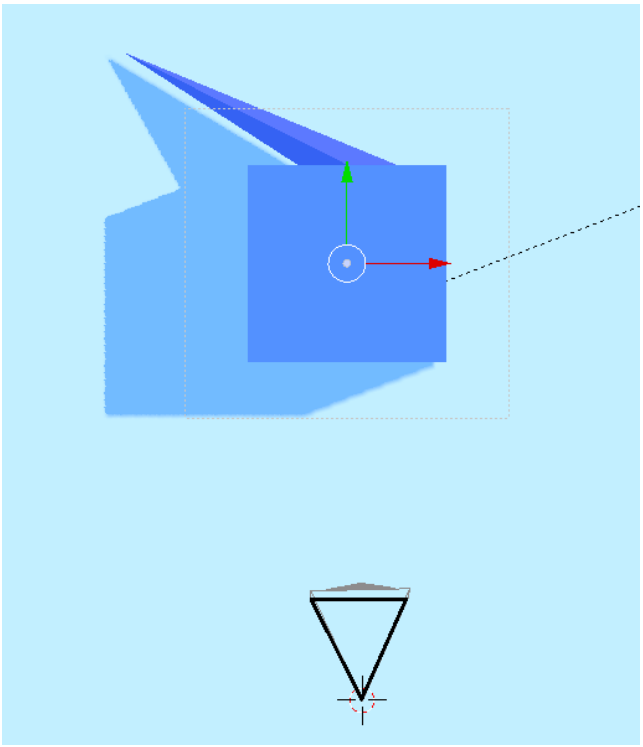

Рис. 3. Скорректированное расположение камеры относительно управляемого объекта.

Перейдём к главному пункту, описывающему процесс создания логической программы [10] управления движением объекта. В данном случае целесообразной является демонстрация модели, сконструированной с помощью логического редактора (рис. 4).

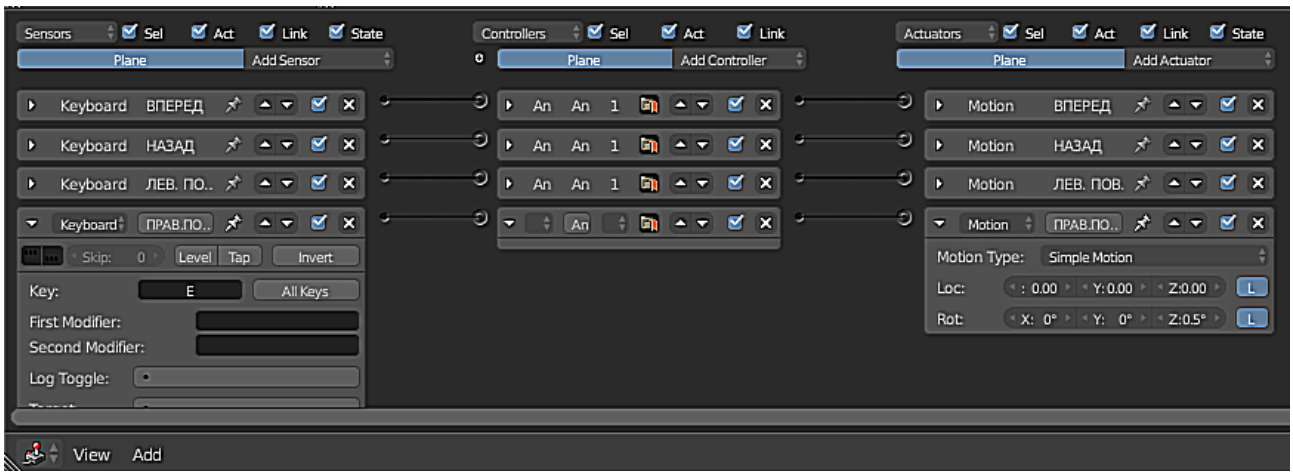

Рис. 4. Модель управления объектом, созданная в логическом редакторе программы UPBGE.

Как видно из рисунка, в редакторе присутствует 3 столбца, в каждом из которых расположены по 4 строки. Столбцы соответствуют типу логических блоков: в первом расположены сенсоры, во втором контроллеры, а в третьем актуаторы. Данные блоки построчно соединены, - это означает однозначность связи: при определённом сигнале сенсора срабатывает определённый контроллер и актуатор.

В строках описаны (сконструированы) следующие логические действия:

1. При нажатии клавиши «W» (сенсор назван «ВПЕРЕД») контроллер отрабатывает логическую команду «And», означающую суммирование (в данном случае актуатор «ВПЕРЕД» осуществляет дополнительное перемещение по оси *y* на 0.1).

2. При нажатии клавиши «S» (сенсор назван «НАЗАД») контроллер отрабатывает логическую команду «And», означающую суммирование (в данном случае актуатор «НАЗАД» осуществляет дополнительное перемещение по оси *y* на отрицательное значение -0.1).

3. При нажатии клавиши «Q» (сенсор назван «ЛЕВ. ПОВОРОТ») контроллер отрабатывает логическую команду «And», означающую суммирование (в данном случае актуатор «ЛЕВ. ПОВОРОТ» осуществляет дополнительное вращение вокруг оси z против часовой стрелки на 0.05°).

4. При нажатии клавиши «E» (сенсор назван «ПРАВ. ПОВОРОТ») контроллер отрабатывает логическую команду «And», означающую

суммирование (в данном случае актуатор «ПРАВ. ПОВОРОТ» осуществляет дополнительное вращение вокруг оси z по часовой стрелке на 0.05°). На рис. 4 данная логическая цепочка показана в развёрнутом виде.

Таким образом, создание модели динамического поведения объекта при нажатии клавиш можно считать законченным. После нажатия кнопки «Start» симулятора, получим картину, изображённую на рис. 5.

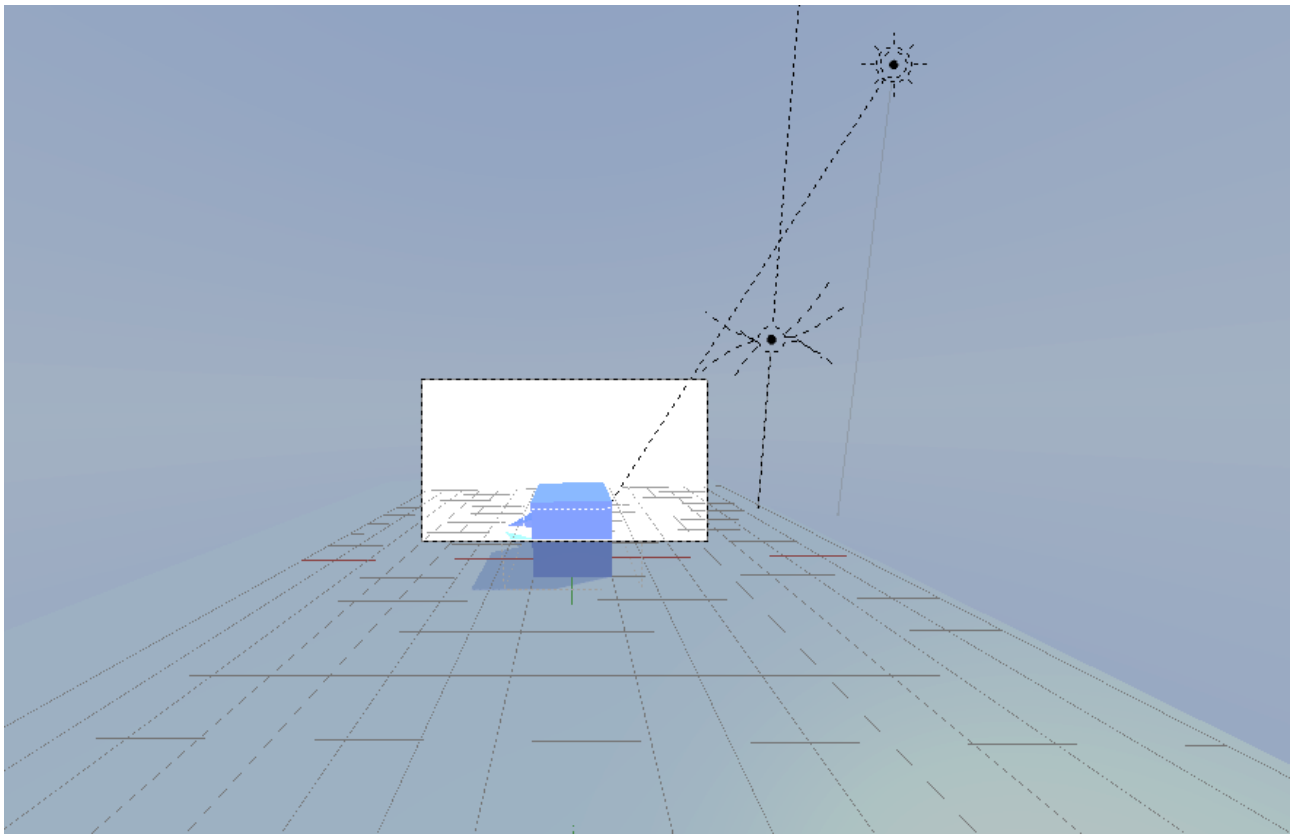

Рис. 5. Картина, формируемая симулятором.

В центре изображения более ярко «выхвачен» участок изображения, отображаемый камерой. В данном случае камера охватывает не весь объект, однако во многих случаях этого вполне достаточно для слежения за его перемещениями. В случае, если пользователю симулятора требуется другой вид, тип и исходное положение камеры легко меняются, как это было продемонстрировано на рис. 3.

Так как в данном случае применено связывание системы координат камеры и объекта, последний никогда не выходит из поля её зрения (то есть

камера перемещается вместе с объектом).

Заключение

Таким образом, цель описываемой работы достигнута: разработан алгоритм создания динамического управляемого пользователем проекта в программе UPBGE, характеризуемого высокой степенью универсальности для создания любых подобных приложений.

На наглядном примере показаны все стадии использования созданной методики, а также многие применяемые при этом встроенные в UPBGE средства, важнейшими из которых являются средства создания логических связей между элементами системы «сенсоры-контроллеры-актуаторы». Также разобрано применение некоторых важных настроек создаваемой виртуальной сцены [11], позволяющее читателю, заинтересованному данной технологией, достаточно легко перейти в дальнейшем к рассмотрению более сложных реалистичных сцен.

Представленная работа является тем более важной, что пока в сети интернет и в прочих источниках имеется весьма скудный набор информации по рассматриваемому вопросу. При этом данная статья ставит своей целью дать любому читателю базовую информацию по созданию динамических моделей для симуляторов, игр и прочих современных приложений [12], не вдаваясь в подробности, доступные для понимания лишь специалистам в данной отрасли.

Имея в своём арсенале бесплатную, доступную для освоения и при этом постоянно совершенствуемую программу-движок для логического динамического моделирования сцен Uchronia Project Blender Game Engine, и созданную авторами статьи методику её пошагового применения, пользователь сможет существенно пополнить свои знания и вывести исследовательскую работу на качественно новый уровень.

Библиографический список:

1. Nasonov D., Ilichev V., Raevsky V. The experimental study of elastic-hysteresis properties of rubber elements of sleeve-pin couplings. //

В сборнике: Proceedings of the 52th International JVE Conference in St. Petersburg. 2021. С. 193-197.

2. Евтухова Е.С. Комбинированный алгоритм моделирования движения динамического объекта в ограниченном пространстве. // Технологии инженерных и информационных систем. 2016. № 2. С. 62-70.

3. Ильичев В.Ю. Создание скриптов Python для управления роботами в симуляторе программы FreeCAD. // Заметки ученого. 2021. № 11-1. С. 181-184.

4. Ильичев В.Ю., Ганков М.С. Разработка методики вычисления и визуализации 3D фракталов с использованием программы Blender. // Заметки ученого. 2022. № 11-1. С. 53-58.

5. Курушин Д.С., Долгова Е.В., Файзрахманов Р.А. Моделирование визуальной одометрии мобильного робота в среде Blender Game Engine. // Международная конференция по мягким вычислениям и измерениям. 2018. Т. 1. С. 434-437.

6. Санжаров В.В., Фролов В.А., Волобой А.Г., Галактионов В.А., Павлов Д.С. Система генерации наборов изображений для задач компьютерного зрения на основе фотореалистичного рендеринга. // Препринты ИПМ им. М.В. Келдыша. 2020. № 80. С. 1-29.

7. Кулажанов Т.К., Крученецкий В.З., Сибанбаева С.Е., Вязигин С.В., Серикулова Ж.К. // О роли и месте интеллектуальных компьютерных средств в образовательном процессе. // Известия высших учебных заведений. Технология текстильной промышленности. 2016. № 4 (364). С. 202-207.

8. Ильичев В.Ю. Использование рекурсивных функций для создания фрактальной графики средствами языка Python. // Системный администратор. 2021. № 3 (220). С. 92-95.

9. Ильичев В.Ю. Использование скриптов на языке Python для управления роботами в симуляторе V-Rep. // Заметки ученого. 2021. № 10. С. 57-60.

10. Антипин А.Ф., Антипина Е.В. Моделирование и анализ программ многомерных интервально-логических регуляторов. // Программные продукты и системы. 2019. № 4. С. 744-749.

11. Панамарев Г.Е., Савельев В.Г., Кудинов А.Н. Технология создания интерактивных трехмерных сцен для виртуальных морских тренажеров. // Эксплуатация морского транспорта. 2018. № 4 (89). С. 148-152.

12. Эрмухамедова Ш., Абдужабаров Н. Создание симуляторов на основе современных информационных технологий. // Теория и практика современной науки. 2019. № 2 (44). С. 387-393.