Так как конечный документ (акты на выполнение работ, наряды) составляется непосредственно для исполнителей, то есть монтеров, удобнее было бы использовать программное обеспечение, которое позволило бы объединить в один документ различные типы заявок. Программное обеспечение должно удовлетворять следующим требованиям:

– ручное заведение заявки;
– просмотр заявки;
– добавление комментариев к заявке;
– наличие базы знаний по ранее возникавшим проблемам и ошибкам;
– поиск заявки;
– установление сроков выполнения.

Для разработки системы использовался графический язык моделирования общего назначения UML [2]. На диаграмме классов представлена совокупность статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения. Иными словами, диаграмма классов предназначена для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Разработанная диаграмма классов показывает, что в главном меню имеются три вкладки, одна из них – активная. Также в главном меню имеются кнопки, при нажатии на которые осуществляется просмотр заявок для выбранной зоны (рисунок 1).

На разработанной диаграмме прецедентов отображены возможности оператора по работе с программой: заведение заявки трех типов (установка, повреждения и обслуживание) и просмотр журнала заявок с возможностью экспорта информации в Microsoft Excel.

При разработке системы учитывалось, что она должна оптимизировать процесс учета заявок, а значит должна работать быстро, безотказно и интерфейс должен быть простым и интуитивно понятным для пользователя.

Система реализована с использованием следующих программно-технических средств: СУБД MS Access, Visual C# 2020. Средой разработки выбрана Visual Studio 2020.

Разработанный программный продукт поддерживается 32-битными и 64-битными операционными системами, поэтому его можно ввести в эксплуатацию практически на любом персональном компьютере. Разработанная система позволяет достигнуть сокращения времени, необходимого для обработки заявок различного типа, поступающих на АТС районного узла электросвязи. В связи с этим повышается производительность труда, снижаются затраты предприятия и упрощается процесс обработки заявок.

Список использованных источников

1. Бизнес-сеть Hinetics [Электронный ресурс]. – Режим доступа: http://www.itsmonline.ru/helpdesk/. – Дата доступа: 21.02.2024.

2. Леоненков, А. В. Нотация и семантика языка UML : электронная книга / А. В. Леоненков. – Интернет университет информационных технологий, 2016. – 205 с.

УДК 004.942

Р. И. БОРДАК, А. П. САФРОНОВ

УО «Мозырский государственный педагогический университет им. И. П. Шамякина» (г. Мозырь, Беларусь)

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ ЯДРА СО СТАТИЧНЫМ ОБЪЕКТОМ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ UNITY

При решении задач различного рода, когда необходимо проводить большое количество вычислений и при этом наглядно визуализировать результат работы, применяют различные способы отображения моделей.

В данной статье рассматривается моделирование движения и взаимодействия ядра со статичным объектом, которое будет проводиться на платформе Unity.

Преимуществом данной платформы является ее наглядность и широкая возможность изменения как начальных параметров, так и различных параметров, влияющих как на динамичные, так и на статичные объекты, что в свою очередь может значительно сказаться на конечном результате [1].

Первым шагов в моделировании движения ядра необходимо создать базовую физическую модель (задание массы, размера и других физических свойств). Для этого в Unity используется компонент Rigidbody, что добавляет физические свойства к объектам [2].

Помимо физических свойств объекта, нужно учитывать силы, которые будут воздействовать на ядро (сила тяжести, гравитационные силы и другие). В Unity для описания таких сил используется метод AddForce, который помогает задать направление и величину силы.

Для взаимодействия объектов используются скрипты языка C# и так называемые «коллайдеры» [2]. Они позволяют объектам взаимодействовать с окружающим миром и другими объектами в модели.

Коллайдеры определяют границы и форму объекта, а также контролируют его взаимодействие с другими объектами. Отсюда и выскакивают определенного рода проблемы моделирования различных явлений на платформе Unity.

Unity использует физический движок для обработки столкновений. Проблема, когда Unity не обрабатывает столкновение снарядов коллайдеров, может возникнуть по нескольким причинам [1]:

1. Скорость и физика.

Если один из объектов движется с очень высокой скоростью, это может привести к проблемам с обработкой столкновений и конечная картина может быть неверно отображена. Важно учесть, что физика в Unity использует не только координаты (X, Y, Z), но также временной вектор. Поэтому при перемещении объектов рекомендуется использовать физические методы, такие как MovePosition, с учетом времени и скорости.

2. Типы коллайдеров.

В Unity есть различные типы коллайдеров. Простые и для 3D-объектов, а также для 2D-объектов. Можно применять любой из них или сразу комбинацию, однако примитивные коллайдеры могут не справляться с поворотами и неоднородным масштабированием объектов.

3. Mesh Colliders.

Если точная форма объекта не соответствует примитивным коллайдерам, вы можете использовать Mesh Colliders для 3D-объектов. Они точно соответствуют форме меша объекта. Однако они более ресурсоемки.

4. Статичные и динамичные коллайдеры.

Статичные коллайдеры не имеют компонента Rigidbody и представляют неподвижные элементы сцены. Динамичные коллайдеры связаны с Rigidbody и могут двигаться. При перемещении динамичных коллайдеров рекомендуется использовать физические методы, чтобы избежать проблем с производительностью.

Чтобы избежать описанных выше проблем, можно воспользоваться следующими советами:

- тщательно настраивать параметры физической модели;
- подбирать коллайдеры исходя из своих задач;
- экспериментировать с временными шагами для достижения оптимальной производительности и точности;
- при необходимости применять специальные техники, такие как интерполяция или предсказание столкновений.

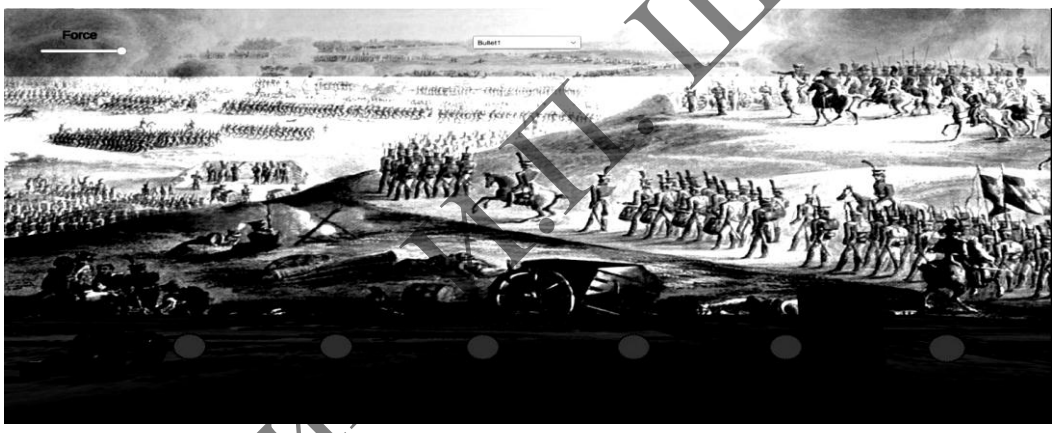

Рисунок 1 – При высоких скоростях снаряда столкновение не обработается

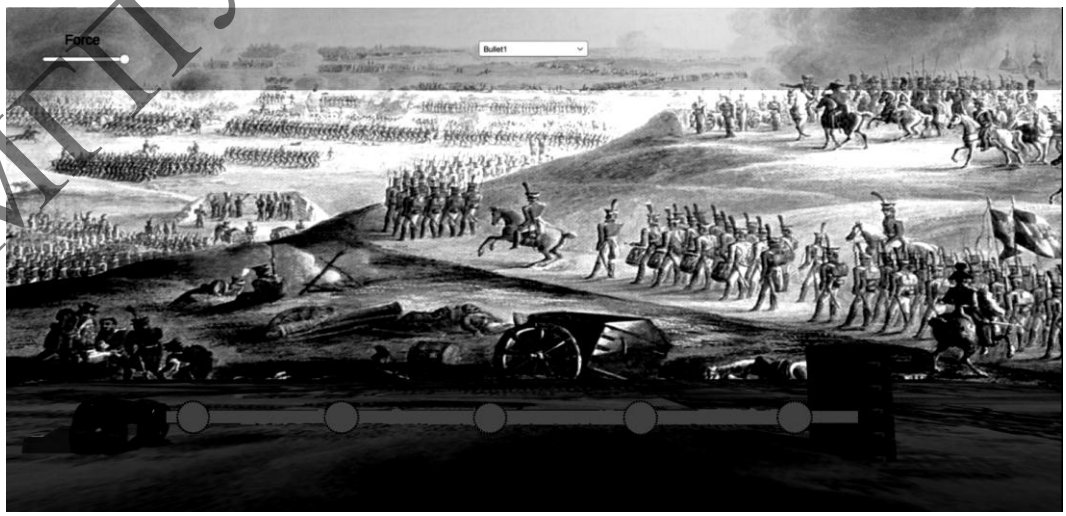

Рисунок 2 – Столкновение будет обрабатываться на всей траектории движения снаряда

Подводя итоги, можно сказать, что, может, сама платформа Unity и имеет определенного рода недостатки, но с ее помощью можно создавать реалистичные и точные симуляции, которые могут быть использованы в различных областях науки и техники.

Список использованных источников

1. Ларкович, С. Н. Справочник Unity. Кратко, быстро, под рукой / С. Н. Ларкович. – СПб. : Наука и техника, 2020. – 288 с.

2. Руководство по Unity // Unity Documentation [Электронный ресурс]. – 2024. – Режим доступа: https://docs.unity3d.com/Manual/index.html. – Дата доступа: 25.02.2024.

УДК 539.12

А. В. БУРЫЙ, А. В. ИВАШКЕВИЧ

Институт физики им. Б. И. Степанова Национальной академии наук Беларуси (г. Минск, Беларусь)

НЕРЕЛЯТИВИСТСКАЯ ЧАСТИЦА СО СПИНОМ 2 В МАГНИТНОМ ПОЛЕ

Задача о частице в магнитном поле является классической для квантовой механики. Первыми были решены уравнение Шредингера и релятивистское уравнение Дирака [1–3]. Значительно позже были найдены решения релятивистского уравнения Даффина – Кеммера для частицы со спином 1, в том числе при наличии у векторной частицы дополнительных характеристик: поляризуемости, аномального магнитного и электрического квадрупольного моментов, а также при учете геометрии 3-мерных пространств постоянной кривизны [4–7].

В настоящей работе мы обращаемся к исследованию в магнитном поле уравнения для частицы со спином 2. Исходным является введенное Ф.И. Федоровым [8] матричное 39-компонентное уравнение 1-го порядка; см. также недавние работы [9, 10]. В частности, в [10] были найдены решения этого уравнения во внешнем однородном магнитном поле; анализ оказался достаточно сложным, в частности, релятивистские спектры энергии получаются как корни алгебраического уравнения 7-го порядка, 5 корней дают физически интерпретируемые спектры энергий. К сожалению, их анализ возможен только численными методами. В работе [11] из 39-компонентного матричного уравнения Федорова было выведено нерелятивистское уравнение, которое напоминает уравнение для нерелятивистской частицы со спином 1/2, но при этом волновая функция имеет пять компонент. В настоящей работе построены решения этой более простой системы уравнений с учетом внешнего магнитного поля.

В [11] было выведено следующее нерелятивистское уравнение для частицы со спином 2 (см. также [9])

$$iD_0\Psi = -\frac{1}{2M}(D_1^2+D_2^2+D_3^2)\Psi - \frac{ie}{2M}(F_{23}S_1+F_{31}S_2+F_{12}S_3)\Psi; \tag{1}$$

волновая функция имеет 5 компонент, явный вид спиновых матриц следующий:

$$\Psi = \begin{vmatrix}\psi_1\\ \psi_2\\ \psi_3\\ \psi_4\\ \psi_5\end{vmatrix}, S_1 = \begin{vmatrix}0&0&-1&0&0\\0&0&0&-2&0\\1&0&0&0&0\\0&1&0&0&-1\\0&0&0&2&0\end{vmatrix}, S_2 = \begin{vmatrix}0&0&0&1&0\\0&0&0&0&0\\0&1&0&0&2\\-1&0&0&0&0\\0&0&-2&0&0\end{vmatrix}, S_3 = \begin{vmatrix}0&-2&0&0&-1\\2&0&0&0&0\\0&0&0&-1&0\\0&0&1&0&0\\0&0&0&0&0\end{vmatrix};$$

символ D_a обозначает удлиненную производную $D_a = \partial_a + ieA_a, a = 0,1,2,3$. Внешнее однородное магнитном поле будем ориентировать вдоль оси z: $B = (0,0,B)$.

Удобно совершить линейное преобразование над волновой функцией, чтобы матрица S_3 стала диагональной:

$$\Psi' = U\Psi, \quad U = \begin{vmatrix}2i&2&0&0&1\\0&0&i&1&0\\0&0&0&0&1\\0&0&-i&1&0\\-2i&2&0&0&1\end{vmatrix}, \quad US_3U^{-1} = \begin{vmatrix}-2i&0&0&0&0\\0&-i&0&0&0\\0&0&0&0&0\\0&0&0&i&0\\0&0&0&0&2i\end{vmatrix} = S_3'. \tag{2}$$