Для работы с объектом используются указатели на их интерфейс, через которые и вызываются методы. Аналогичным образом происходит и создание объектов, посредством вызова метода через интерфейс.

DirectX основан на *Component Object Model* (*COM*) – объектно-ориентированной модели программирования, использующей несколько технологий. *COM* позволяет работать с объектно-ориентированной моделью любого языка, которые его поддерживают. Для работы с *DirectX* через диспетчера пакетов *NuGet* необходимо установить библиотеку *SharpDX* – ядро для всех сборок *SharpDX*, а также библиотеки *SharpDX.Desktop, SharpDX. Direct2D1*, *SharpDX.DirectInput, SharpDX.DXGI*, *SharpDX.Mathematics*, позволяющие работать со средствами *DirectX*. Приложение запускается в одном окне. Окно рендеринга библиотеки *SharpDX* воспроизводится через элемент управления *WPF*, а информационная статистика выводится в самом *WPF*.

М. А. Голубев, Е. В. Комракова
(*ГГТУ им. П. О. Сухого, Гомель*)

ОПТИМИЗАЦИЯ МАТРИЦЫ СТОЛКНОВЕНИЙ

Физическая система поддерживает матрицу столкновений, определяющую пары объектов, которые могут соприкасаться. Объекты, не включенные в эту матрицу, автоматически игнорируются физической системой при обнаружении перекрытия их объемов.

Матричная система определения столкновений использует в своей работе систему слоев *Unity*. Матрица хранит все возможные комбинации слоев, а установка флага означает, что на этапе определения столкновений будут проверяться коллайдеры обоих этих слоев. Также нельзя заставить отреагировать на столкновение только один из двух объектов, то есть если один слой может сталкиваться с другим слоем, они оба должны реагировать на столкновения.

Общее число слоев в проекте не может превышать 32, поэтому необходимо продумать распределение объектов между слоями, которое будет использовано во всем проекте. Если по какой-то причине 32 слоев окажется недостаточно, можно попробовать найти способы повторного использования слоев или удалять слои, ставшие ненужными. Все прочие пары слоев или объектов просто игнорируются физи-

ческим движком, что делает этот механизм важным средством уменьшения нагрузки.

В игре сведено к минимуму количество проверок возможных столкновений между объектами. Так как источники энергии (*Powerups*) может собирать только игрок (*Player*), нет необходимости отслеживать столкновения между источниками энергии и объектами из других слоев. С другой стороны, нет смысла отслеживать столкновения снарядов с объектами, стреляющими ими, что отражено в исключении столкновений снарядов врагов (*Enemy Projectiles*) с самими врагами (*Enemies*) и столкновений снарядов игрока (*Player Projectiles*) с самим игроком (*Player*) [1, с. 613].

Необходимо проверить все комбинации слоев в матрице столкновений на соответствие логике, чтобы убедиться, что драгоценное время не тратится на ненужные проверки между неподходящими парами объектов.

Литература

1 Бонд, Д. Unity и C#. Геймдев от идеи до реализации / Д. Бонд. – СПб. : Питер, 2016. – 613 с.

Е. Д. Григоренко, Н. Б. Осипенко
(*ГГУ им. Ф. Скорины, Гомель*)

ПРИЛОЖЕНИЕ «КАДЕТСТВО» НА ЯЗЫКЕ C#

Современный мир уже невозможно представить без разнообразного количества информационных технологий. Они повсеместно используются во всевозможных сферах деятельности человека, в том числе и в процессах воспитания и обучения. Образовательный процесс тесно связан и с производством, и с обменом, и с хранением, и с использованием различной информации. Зачастую она хранится на бумажных носителях, что ведет к сложности поиска необходимых данных.

Работа посвящена описанию автоматизированной системы управления образовательным процессом в ГУО «Гомельское кадетское училище». Приложение позволяет создать единое пространство в виде электронной базы данных с информацией об училище, сотрудниках, учащихся и их родителях, учебном расписании, успеваемости