Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Григорьев Роман Витальевич

Реализация алгоритмов для системы геометрических частиц в игровом движке Saber3D

Бакалаврская работа

Научный руководитель:
доц. каф. СП, к. т. н. Литвинов Ю. В.

Консультант:
инженер-программист ООО “Сабер Интерактив СГС” Гурьев В. А.

Рецензент:
ведущий инженер-программист ООО “Сабер Интерактив СГС” Сладков Д. В.

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Grigorev Roman

Implementation of algorithms for the geometric particle system in the Saber3D game engine

Bachelor's Thesis

Scientific supervisor:
C.Sc. Docent Yurii Litvinov

Consultant:
Graphics Programmer, Saber Interactive, Vasilii Gurev

Reviewer:
Lead Graphics Programmer, Saber Interactive, Denis Sladkov

Saint-Petersburg
2021

Оглавление

Введение

Системы частиц уже десятки лет являются важной частью видеоигр и компьютерной графики в целом. Первая публикация на тему систем частиц была написана после завершения работы над визуальными эффектами для кинофильма «Star Trek II» [9]. В ней Уильям Ривз описал основные операции движения и основные данные, представляющие частицу. Системы частиц используют множество частиц, чтобы моделировать объекты или явления с нечеткими границами. В качестве частиц могут использоваться точки, треугольники, текстуры и трёхмерные модели. В течение определенного периода времени частицы порождаются, некоторые их параметры изменяются согласно заданному сценарию, после чего частицы удаляются из системы. При помощи частиц становится возможным, например, моделировать огонь, дым, взрывы, воду, вращающиеся галактики, волосы, облака и другие явления. Системы, использующие в качестве частиц трехмерные модели, называются геометрическими.

Уже около 20 лет компания Saber Interactive ведет работу над собственным игровым движком Saber3D [14], [10]. Впервые он использовался в игре Will Rock (2003 год), и с тех пор постоянно обновляется и развивается для использования в современных продуктах. В процессе работы над обновлением различных компонентов движка была создана новая система геометрических частиц. Создание и настройка систем геометрических частиц движка Saber3D выполняется в редакторе игровых уровней — программе, позволяющей, например, создавать игровые карты, размещать на них модели и настраивать освещение. Арт-отделу необходим максимальный контроль над системой, но в данный момент в редакторе нет возможности, например, изменять размер частиц вдоль осей неравномерно, нельзя сделать так, чтобы частицы во время полета были ориентированы в соответствии с вектором направления. В редакторе уровней также отсутствуют инструменты, позволяющие переопределять свойства материала частиц. Кроме того, в редакторе не хватает функциональности, помогающей производить отладку частиц. Итак, на

данный момент система геометрических частиц движка Saber3D обладает недостаточной функциональностью, из-за чего становится невозможным использовать ее для создания сложных игровых эффектов.

Таким образом, требуется расширить текущую функциональность системы геометрических частиц движка Saber3D. Для этого необходимо решить ряд задач, связанных с настройкой свойств материала, ориентации частицы в пространстве и ее размеров, а также с выводом отладочной информации. Наличие в редакторе уровней перечисленной функциональности позволит арт-отделу полноценно использовать систему частиц при создании сложных игровых эффектов. Стоит также отметить, что помимо программной реализации новой функциональности необходимо добавить элементы взаимодействия пользователя с новой функциональностью в интерфейс управления системой частиц в редакторе игровых уровней.

1. Постановка задачи

Целью данной работы является реализация новой функциональности для системы геометрических частиц игрового движка Saber3D.

Для достижения цели были сформулированы следующие задачи.

- Изучить реализацию существующей системы геометрических частиц в движке Saber3D.
- Выявить участки кода, нуждающиеся в переработке, и провести их рефакторинг.
- Реализовать для существующей системы геометрических частиц движка Saber3D следующую функциональность:
  - неравномерное изменение размера частиц;
  - изменение ориентации частиц по вектору направления полета;
  - отслеживание текущей статистики частиц;
  - переопределение свойств материала частиц;
  - полезные функции из существующих решений.
- Интегрировать разработанный код в основную ветвь исходного кода движка Saber3D и апробировать новую функциональность.

2. Обзор

Целью обзора является ознакомление с существующими подходами к реализации систем частиц. В частности, чтобы повысить качество решения поставленных задач, необходимо исследовать современную функциональность, предоставляемую пользователям для манипуляций частицами.

Системы частиц бывают двухмерные и трехмерные (геометрические). В первом варианте частицы представляют собой 2D текстуру (биллборд), которая всегда ориентирована лицевой стороной к камере пользователя [1] [4]. Во втором варианте частицы — это трехмерные модели.

За порождение частиц отвечает эмиттер — сущность, с помощью которой можно задать параметры порождения частиц (скорость, с которой объекты порождаются, их количество, направление, в котором они излучаются), начальные свойства частиц (время жизни, размер, вес, цвет, текущая позиция, скорость) [9] и то, как эти свойства будут изменяться с течением времени [5]. Свойство “время жизни” позволяет настраивать свойства, зависящие от времени, а также определять, когда частицу нужно удалить из системы.

Если время жизни становится равным нулю, частица уничтожается. Однако определять конец жизни можно и другими способами. Например, частица может уничтожаться, когда ее яркость опустится ниже некоторого значения. Кроме того, можно уничтожать частицы, если они ударились о поверхность или покинули некоторый регион пространства.

Работа системы частиц начинается с этапа симуляции. Этот этап отвечает за создание новых частиц и инициализацию их свойств начальными значениями, обновление свойств ранее порожденных частиц в зависимости, например, от прошедшего с момента порождения времени. Кроме того, выполняются необходимые действия для частиц, у которых истекло время жизни [12].

Не менее важным является этап сортировки для двухмерных ча-

стиц. Система частиц, моделирующая костер, не должна быть видна игроку, если перед ней находится система частиц, моделирующая дымовую завесу. Некоторые типы смешивания, такие как альфа смешивание, не являются коммутативными, и поэтому визуальный результат будет зависеть от порядка, в котором частицы обрабатывались. Для решения этой проблемы частицы сортируются в порядке удаления от камеры.

В дальнейшем частицы должны будут пройти этапы обработки видимости (опционально) и рендеринга.

2.1. Обзор существующих решений в области систем частиц

2.1.1. Система частиц в Unity

Unity — современный движок для разработки компьютерных игр, позволяющий создавать приложения для более чем 25 различных платформ. Основан на модульной системе компонентов.

Система частиц в Unity состоит из множества модулей, но выделить можно четыре основных [8]:

- главный модуль;
- модуль эмиссии;
- модуль формы;
- модуль рендера.

Главный модуль позволяет настраивать атрибуты частицы, такие как время жизни, начальные скорость, размер, цвет.

Модуль эмиссии имеет два параметра. Первый позволяет управлять количеством частиц, которые испускаются за одну секунду, а второй — на единицу расстояния (один юнит).

Модуль формы определяет форму, согласно которой частицы испускаются. Возможно выбрать одну из нескольких различных форм, каждая из которых имеет свои собственные настройки. Это позволяет создавать частицы в коробке, сфере или даже в определенном пользователем меше — совокупности вершин, рёбер и полигонов, которые составляют один 3D объект.

Помимо основных модулей, существуют также дополнительные, которые позволяют в течение жизни частицы согласно сплайнам изменять такие атрибуты, как размер, цвет, скорость. Интересно, что в Unity предусмотрена возможность присваивать атрибутам частицы случайные значения с помощью двух сплайнов, которые определяют диапазон допустимых значений.

Модуль рендера содержит настройки, управляющие отображением частиц в игре. Например, можно выбрать, рисовать частицу в виде биллборда или в виде 3D модели.

2.1.2. Система частиц в Unreal Engine 4

Unreal Engine — современный игровой движок, разработанный Epic Games. На данный момент активно развивается и является одним из самых популярных игровых движков в мире.

В Unreal Engine 4 есть удобная система под названием Cascade, которая позволяет создавать эффекты, состоящие из модулей, а также позволяет настраивать поведение частиц [13].

Модули используются для контроля свойств частиц, и могут применяться как во время создания частицы, так и во время ее жизни. Модули взаимодействуют друг с другом в зависимости от их расположения в стеке модулей Cascade. Модули Unreal Engine разбиваются на группы. В качестве примера можно выделить группы ускорения, коллизии, цвета, времени жизни, ориентации, поворота, размера, скорости.

Рассмотрим группу размера. В ней находятся пять модулей:

- Initial Size — устанавливает начальный размер частицы во время порождения;

- Initial Size (Seeded) — идентичен Initial Size, но позволяет установить дополнительные случайные параметры;
- Size By Life — масштабирует размер частицы на заданное значение в течение ее времени жизни;
- Size Scale — устанавливает размер частицы равным начальному размеру, умноженному на заданный коэффициент;
- Size By Speed — масштабирует размер частицы на значение, соответствующее некоторой части ее скорости.

Стоит отметить параметр Distribution, который позволяет задать метод, по которому будет изменяться какое-либо свойство частицы. Distribution — группа типов данных, среди которых можно выделить следующие:

- DistributionFloatConstant — значением свойства является константа;
- DistributionFloatUniform — значением свойства является случайное число из диапазона значений;
- DistributionFloatConstantCurve — задает кривую, где координата Y — значение свойства, а координата X — время жизни частицы или эмиттера;
- DistributionFloatUniformCurve — задает две кривые, представляющие собой диапазон значений;
- DistributionVectorConstant — значением свойства является константный вектор (может использоваться, например, для масштабирования частицы по трем осям);
- DistributionVectorUniform — значением свойства является случайный вектор. Для каждой компоненты вектора задается диапазон значений;

- DistributionVectorConstantCurve — аналогично DistributionFloatConstantCurve, однако задаются три кривые, каждая из которых отвечает за значение компоненты вектора;
- DistributionVectorUniformCurve — аналогично DistributionVectorConstantCurve, однако для каждой компоненты вектора задаются две кривые, определяющие диапазон значений.

2.1.3. Система частиц в Source Engine 2013

Source Engine — известный игровой движок, разработанный и поддерживаемый компанией Valve. Является преемником движка GoldSrc [11].

Система частиц в Source состоит из нескольких компонентов [6]:

- рендереры — определяют способ рисования частиц (веревки, спрайты, капли жидкости, модели);
- эмиттеры — определяют количество создаваемых частиц и тип излучения;
- инициализаторы — устанавливают начальное состояние каждой частицы;
- операторы — позволяют изменять частицы после инициализации (изменять прозрачность, цвет, размер, задавать вращение);
- силы — позволяют влиять на движение частицы;
- ограничения — определяют ограничения на движение частиц;
- дочерние системы частиц — другие системы частиц, привязанные к исходной.

Изменять размер частицы в течение времени можно с помощью оператора Radius Scale. У него есть пять основных параметров:

- radius_start_scale — значение масштаба в начале масштабирования;

- radius_end_scale — значение масштаба в конце масштабирования;
- start_time — время начала масштабирования (в секундах);
- end_time — время конца масштабирования (в секундах);
- scale_bias — позволяет ускорить или замедлить применение эффектов Radius Scale.

Возможность изменять параметры частиц с помощью сплайнов не предоставлена.

2.1.4. Система частиц в Amazon Lumberyard

Amazon Lumberyard — современный кросс-платформенный игровой движок, разработанный компанией Amazon [2]. Состоит из нескольких основных систем, которые позволяют, например, работать со звуками, создавать уровни, работать с материалами и системой частиц.

Управление системой частиц осуществляется с помощью редактора частиц. В этом редакторе есть несколько атрибутов-секций, каждая из которых позволяет изменять свойства для частиц. Например, коллизию, освещение, излучение, движение, вращение, размер. В редакторе частиц используется редактор кривых, который позволяет изменять свойства с помощью кривой. Некоторые параметры в атрибутах-секциях содержат дополнительные настройки [7]:

- Random — определяет, насколько значение параметра частицы отклоняется от стандартного значения;
- Strength Over Emitter Life — изменение параметра в течение жизни эмиттера с помощью кривой;
- Strength Over Particle Life — изменение параметра в течение жизни частицы с помощью кривой.

Например, атрибут Size содержит параметры Size X, Size Y, Size Z, каждый из которых позволяет масштабировать частицы вдоль соот-

ветствующей оси как с помощью числового значения, так и с помощью кривых, упомянутых выше.

Стоит отметить, что в движке Lumberyard в геометрической системе частиц возможно выбирать тип излучения 3D моделей. А именно, если основная модель состоит из нескольких подмоделей, то существует возможность излучать как основную модель, так и ее подмодели.

2.1.5. Система частиц в ChilliSource

ChilliSource — это кросс-платформенный игровой движок с открытым исходным кодом [3]. Общая архитектура системы частиц представлена на рисунке 1.

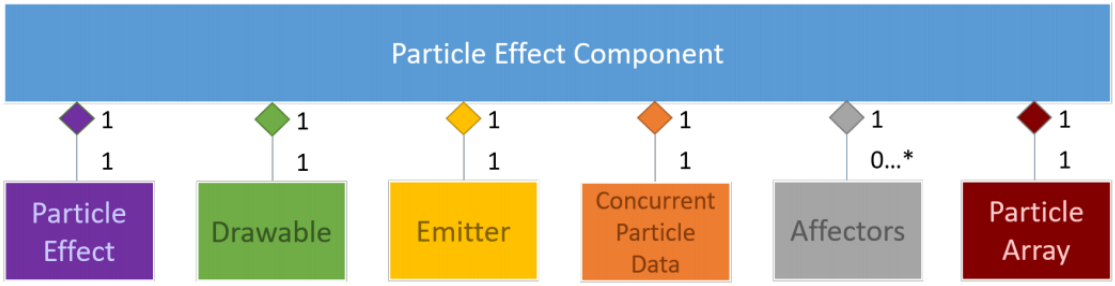

Рис. 1: Архитектура системы частиц в ChilliSource [3]

Применять эффекты к частицам, которые меняются с течением времени, можно с помощью специальных объектов — аффекторов. В исходном коде аффекторы реализованы через базовый класс ParticleAffector, от которого наследуются 4 класса:

- ScaleOverLifetimeParticleAffector — изменяет размер частицы;
- ColourOverLifetimeParticleAffector — изменяет цвет частицы;
- AngularAccelerationParticleAffector — изменяет угловую скорость частицы;
- AccelerationParticleAffector — изменяет скорость частицы.

Изменение вышеперечисленных атрибутов происходит линейно с течением времени. В ChilliSource не предоставлена возможность изменять параметры частиц с помощью сплайнов.

2.2. Обзор системы частиц в движке Saber3D

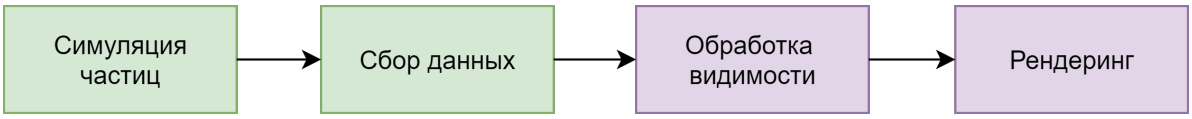

Рис. 2: Порядок обработки системы геометрических частиц

Порядок обработки системы геометрических частиц Saber3D можно разделить на 4 основных этапа, представленных на рисунке 2. На этапе симуляции происходит создание частиц, начальная инициализация их свойств, а также обновление этих свойств на каждом кадре. На этапе сбора данных создаются матрицы трансформации и ограничивающие объемы. Затем на этапе обработки видимости отбрасываются те частицы, которые не видны игроку. На этапе рендеринга происходит передача данных на видеокарту в шейдеры. Шейдеры — это программы, исполняющиеся видеокартой и определяющие визуальное представление всех объектов на сцене.

Система геометрических частиц Saber3D работает с шаблонами — объектами, состоящими из одной модели или нескольких. Пример шаблона из моделей камней представлен на рисунке 3. В роли частиц выступают модели некоторого шаблона. Такой подход позволяет генерировать частицы различного внешнего вида. Частицы порождаются эмиттером. Эмиттер может быть представлен, например, точкой в пространстве или поверхностью. К частицам возможно привязывать другие эмиттеры. Иными словами, эмиттер может испускать дополнительные эмиттеры.

Рис. 3: Различные типы моделей шаблона

Рис. 4: Процесс порождения частиц в редакторе

На рисунке 4 центр основного эмиттера помечен координатными осями справа. Желтым цветом помечены дополнительные эмиттеры, которые испускаются из центра основного эмиттера и порождают сами частицы.

Каждый эмиттер на сцене содержит ссылку на шаблон, который он использует. А каждая частица, излученная эмиттером, содержит номер модели шаблона, которую она использует (например, эмиттер может использовать шаблон из 20 видов камней, а частица хранит номер камня).

2.3. Выводы

В процессе обзора существующих решений было выполнено ознакомление с существующими системами частиц и их функциональностью. К достоинствам Unity, Unreal Engine 4 и Amazon Lumberyard можно отнести удобный интерфейс и множество различных функций

для управления системами частиц. Более того, системы частиц этих движков состоят из множества модулей, которые можно добавлять и удалять. Каждый модуль представляет определенный аспект поведения частицы и содержит только свойства, которые управляют этим поведением. Так как добавляются только модули для необходимого поведения, то не выполняются лишние вычисления ненужных свойств.

Кроме того, три вышеупомянутых движка обладают функциональностью, которая в дальнейшем может оказаться полезной в Saber3D. Например, они позволяют присваивать случайное значение свойствам частиц при помощи диапазона из двух сплайнов. В Amazon Lumberyard возможно изменять параметры в зависимости от времени жизни не только частицы, но и эмиттера. Также стоит обратить внимание на параметр Distribution из Unreal Engine, который позволяет удобно выбирать метод изменения свойств частиц. Кроме того, в Unreal Engine возможно включить отображение координатных осей, которые будут явно указывать ориентацию частицы в пространстве.

Система частиц Source Engine 2013 не обладает важными функциями. Например, не предоставлена возможность изменять настройки частиц при помощи кривых, нет компоненты, отвечающей за управление звуком частиц. Однако Source все еще остается довольно мощным инструментом, позволяющим создавать современные игры. Кроме того, он несложен в освоении.

К недостаткам ChilliSource можно отнести недостаточную функциональность системы частиц. Например, не предоставлена возможность изменять параметры при помощи кривых, нет компоненты, позволяющей настраивать звук для частиц, а также нельзя использовать трехмерные частицы. Однако ChilliSource обладает несложной реализацией, что позволяет использовать его в учебных целях.

В рамках обзора выполнено ознакомление с реализацией и принципами работы системы геометрических частиц движка Saber3D. В процессе изучения исходного кода выявлено несколько участков кода, которые нуждаются в рефакторинге.

3. Рефакторинг

3.1. Разделение пользовательского интерфейса

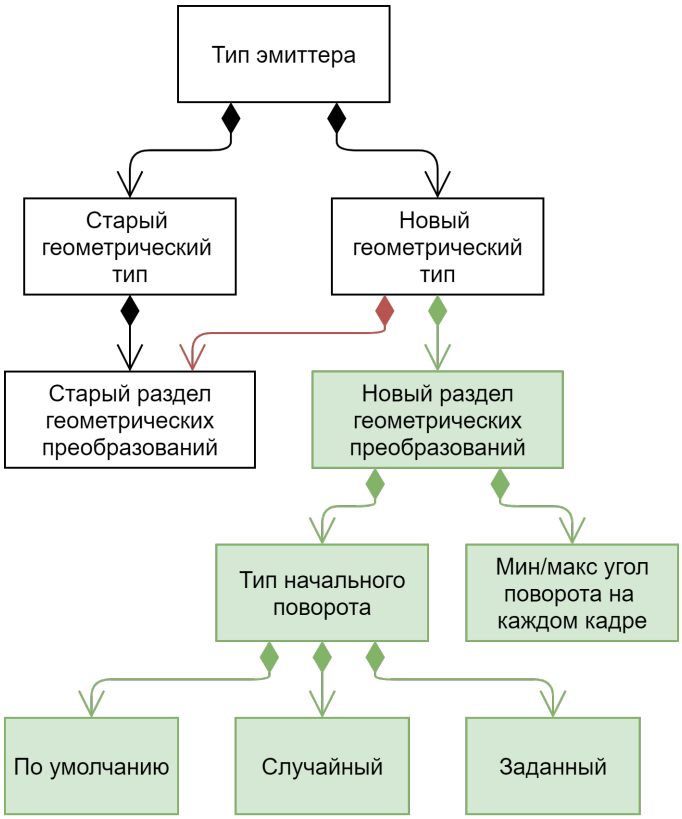

Рис. 5: Красным помечены удаленные элементы, зеленым — добавленные

В текущей реализации системы геометрических частиц есть два типа эмиттеров: первый порождает геометрические частицы, используя старую реализацию системы геометрических частиц, а второй использует новую систему геометрических частиц. Однако оба типа эмиттеров используют одну и ту же секцию-структуру с геометрическими преобразованиями.

Это является проблемой, поскольку новая функциональность добавляется только для новой системы геометрических частиц и не должна затрагивать старую. Таким образом, было принято решение добавить новую секцию с геометрическими преобразованиями, как на рисунке 5. В целях дальнейшего улучшения организации интерфейса в отдельные секции были вынесены элементы, которые позволяют настраивать

различные типы поворотов частиц при порождении, а также задавать минимальный и максимальный угол поворота частиц на каждом кадре после порождения.

3.2. Нарушения закона Деметры

В процессе исследования реализации этапа симуляции системы геометрических частиц было выявлено множество участков кода, на которых получение данных из пользовательского интерфейса выполнялось напрямую. Например, так выполнялось получение данных о размере частицы: pData->emitterType.geometry.transforms.size.value.

Это является проблемой, поскольку пользовательский интерфейс постоянно изменяется, и все такие цепочки тоже приходится исправлять в программе. Кроме того, это существенно увеличивает объем кода и усложняет его чтение. Для решение данной проблемы все такие цепочки были помещены в геттеры, которые в свою очередь находятся в классе, описывающем эмиттер.

3.3. Удаление старой реализации

После реализации функциональности, описанной в данной работе, было принято решение отказаться от старой системы геометрических частиц. Для этого необходимо удалить из пользовательского интерфейса и кода движка всю реализацию, которая не используется в новой системе геометрических частиц.

В основном рефакторингу подвергся код, относящийся к этапу симуляции: большая часть изменений была внесена в реализации классов частиц и эмиттеров. Всего было проверено около 12000 строк кода. Кроме того, в классах, описывающих частицы и эмиттеры, был переименован целый ряд полей и методов, поскольку их прежние названия были неинформативными.

4. Реализация новой функциональности

4.1. Неравномерное изменение масштаба частицы

Одной из основных задач является предоставление возможности пользователю изменять масштаб частицы неравномерно по трем осям ее локальной координатной системы. На рисунке 6 слева изображена частица в стандартном масштабе, а справа изображена частица, вытянутая вдоль своей оси Z.

Рис. 6: Частица справа увеличена вдоль своей оси Z

В пользовательский интерфейс редактора уровней было добавлено три секции, как на рисунке 7. Каждая из них отвечает за изменение масштаба частицы вдоль осей локальной координатной системы X, Y и Z соответственно.

Задавать новый размер можно в виде константы при помощи двух текстовых полей, которые отвечают за минимальный и максимальный возможный масштаб порождаемой частицы. Кроме того, задать размер можно при помощи сплайнов. Горизонтальная ось X сплайна — это время жизни частицы, она принимает значения от 0 до 1. Вертикальная же ось Y — это изменение масштаба вдоль соответствующей оси локальной координатной системы частицы. Таким образом, размер ча-

стицы будет постепенно изменяться с течением времени в соответствии с заданным сплайном. В старой реализации системы геометрические частицы возможно было масштабировать только равномерно.

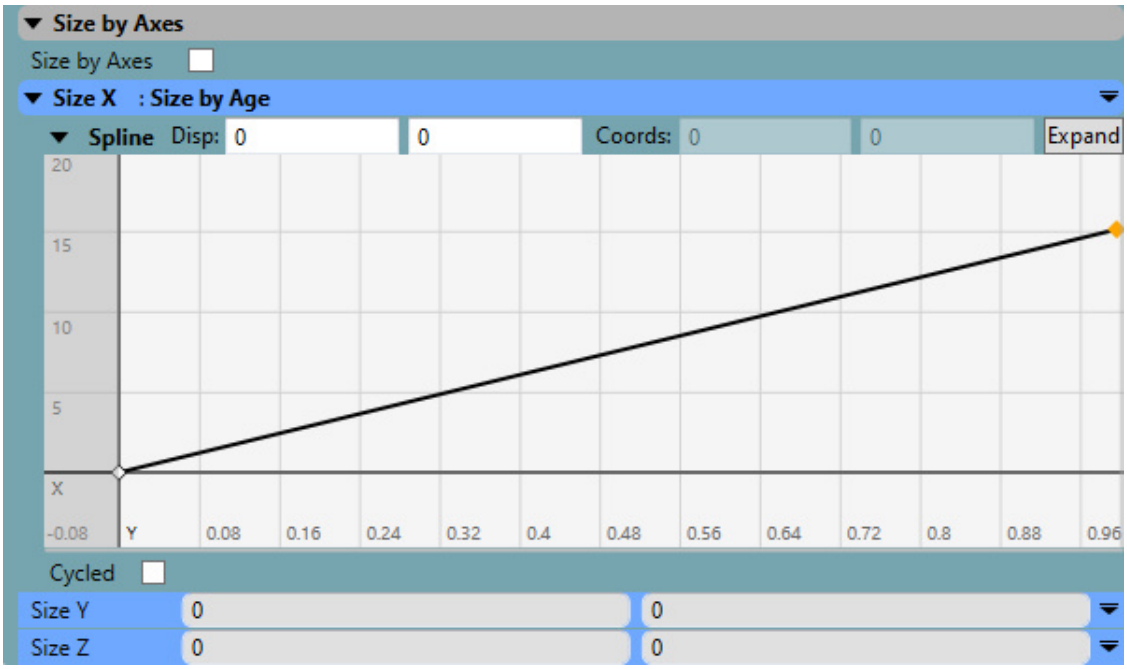

Рис. 7: Секции для изменения масштаба частицы по трем осям

Все порожденные частицы на сцене хранят свой масштаб в виде вектора из трех значений типа float. Этот вектор является полем в структуре, описывающей частицу. На этапе симуляции это поле инициализируется при создании частицы, а обновляется — на каждом кадре, пока у частицы не закончится время жизни. Затем значения из этих векторов, содержащих масштабы, будут записаны в матрицы трансформации, которые создаются для всех порожденных частиц на этапе сбора данных.

4.2. Изменение ориентации по вектору направления

Одной из важнейших функций системы геометрических частиц является возможность выравнивать частицы по их вектору направления полета. Этот вектор инициализируется при порождении частицы и затем обновляется на каждом кадре. На рисунке 8 слева частицы порождаются в своей стандартной ориентации. На рисунке справа частицы

выровнены по своим векторам направления полета.

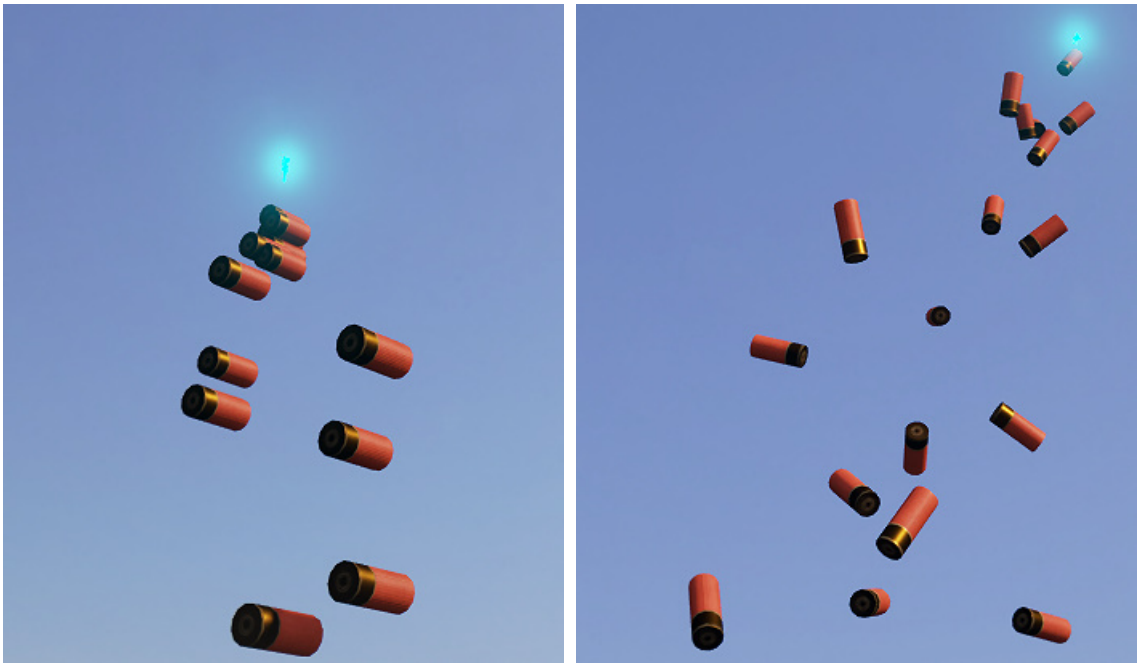

Рис. 8: Частицы справа выровнены по своим векторам направления полета

Функцию можно включать и отключать при помощи галочки в пользовательском интерфейсе. Для реализации поставленной задачи на этапе сбора данных для всех частиц, у которых включена данная функция, строится локальная координатная система частицы.

Изначально в качестве вектора Z принимается вектор направления полета. Затем, при помощи векторного произведения векторов Z и (0, 1, 0) получается вектор X. И, наконец, вновь при помощи векторного произведения векторов Z и X получается вектор Y. После чего создается матрица, у которой в первой, второй и третьей строке находятся только что полученные векторы X, Y, Z соответственно. Эта матрица используется в дальнейшем на этапе сбора данных как начальная матрица трансформации. Если функция выравнивания частиц по вектору направления отключена, то в качестве начальной матрицы трансформации берется единичная матрица.

4.3. Отслеживание текущей статистики частиц

Для того, чтобы можно было удобно отслеживать текущее состояние системы геометрических частиц на сцене, была поставлена задача реализовать таблицу со статистикой. На рисунке 9 приведен пример таблицы для сцены с двумя эмиттерами, первый из которых порождает гильзы, а второй — камни.

Рис. 9: Статистика частиц

Статистика выводит как общую информацию, так и информацию, относящуюся к каждому эмиттеру в отдельности. В самой верхней строке указано общее количество различных шаблонов, которые используются эмиттерами, эмиттеров и порожденных эмиттерами частиц на сцене. Ниже в таблице в каждой строке указана информация о некотором эмиттере на сцене: номер и название эмиттера, номер и название используемого шаблона, количество порожденных частиц, количество моделей в шаблоне и количество используемых моделей шаблона в текущий момент на сцене.

Структура, которая хранит информацию для статистики, очищается на каждом кадре перед началом выполнения этапа симуляции. А затем, во время выполнения этапа сбора данных, заполняется.

4.4. Переопределение свойств материала частиц

Внешний вид частиц определяют материалы. Материал — это совокупность всех характеристик визуального представления поверхностей в движке. Работа с системой частиц осуществляется в редакторе игровых уровней, а настройка свойств материала и добавление в него дополнительных текстур выполняется в отдельной программе. По этой причине арт-отделу потребовалась возможность переопределять некоторые свойства материала частиц сразу в редакторе игровых уровней.

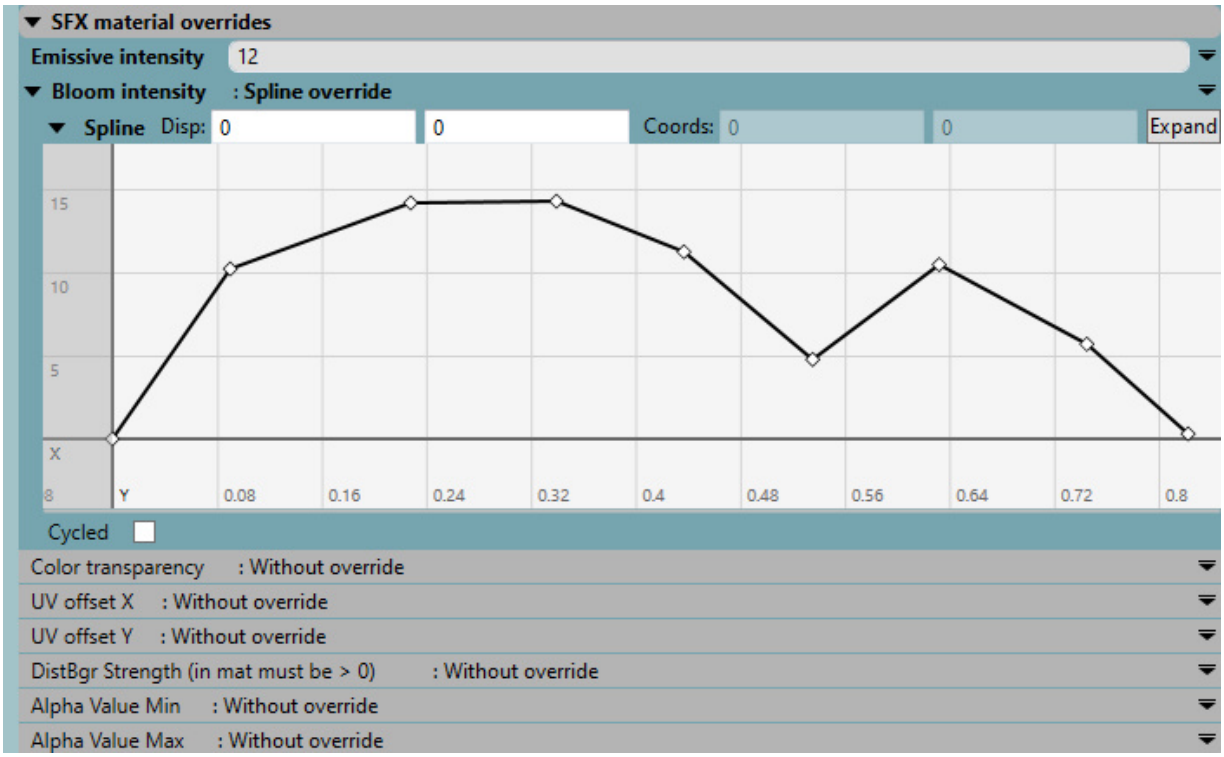

Рис. 10: Интерфейс для переопределения свойств материала

В общей сложности необходимо переопределять 8 свойств:

- интенсивность эмиссии (яркость объекта);
- интенсивность блума (яркость ореола рассеивания света вокруг объекта);
- прозрачность цвета;
- смещение текстурных координат вдоль осей X и Y;

- сила искажения фона;
- минимальное и максимальное значение альфы.

В пользовательский интерфейс редактора уровней была добавлена секция “SFX material overrides”, в которой возможно выбрать свойство и задать его новое значение в виде константы или сплайна, как на рисунке 10. Ось X сплайна принимает значения от 0 до 1 и представляет собой время жизни частицы. А ось Y представляет значения переопределяемого свойства. Таким образом, сплайн позволяет изменять соответствующее свойство материала с течением времени.

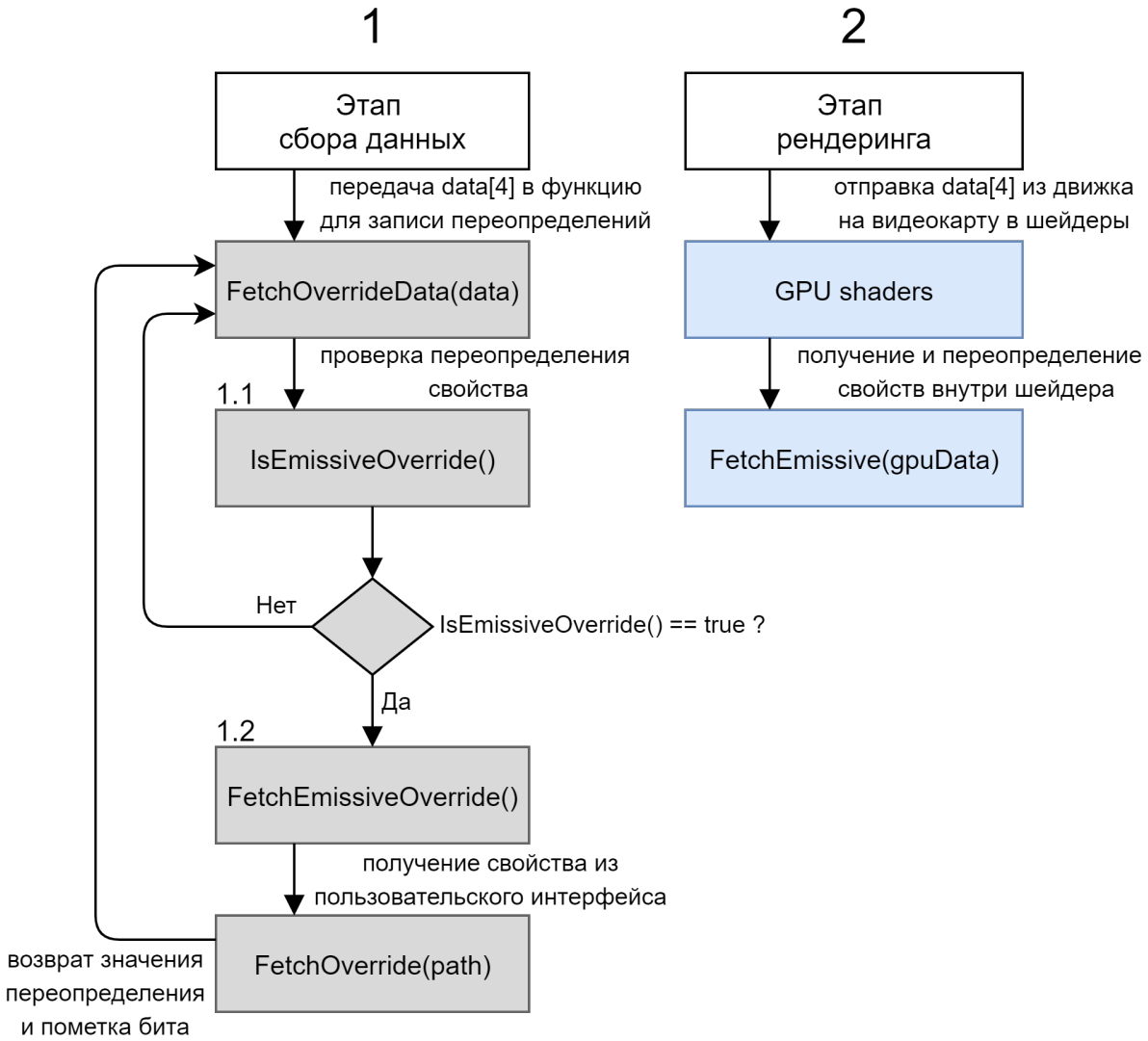

Рис. 11: Схема алгоритма для переопределения свойств материала

На этапе сбора данных, как показано на рисунке 11 под пунктом 1, для каждой частицы в функцию FetchOverrideData() передается пустой массив из 4 переменных типа float. В этой функции в переменные массива будут записаны значения переопределенных свойств материала частицы. Для каждого свойства материала выполняется проверка на его переопределение, как показано в пункте 1.1. Если свойство было переопределено, то в соответствующую переменную массива будет записано переопределенное значение, как показано в пункте 1.2. Кроме того, в функции FetchOverrideData() выполняется "сжатие" полученных значений переопределений: данные, изначально имеющие тип float в 32 бита, упаковываются в 8 бит или в 16 бит. Кроме того, первые 8 бит первой переменной массива хранят информацию о том, было ли переопределено соответствующее свойство.

В дальнейшем на этапе рендеринга, как показано на рисунке 11 под пунктом 2, значения переопределений для всех частиц будут переданы в шейдеры. В шейдерах вместо оригинальных значений свойств из материала будут использоваться новые переопределенные значения, если последние были заданы.

На видеокарте значения переопределений одной частицы хранятся в одном регистре. Регистры состоят из 4 переменных типа float. По этой причине массив имеет 4 переменные. Кроме того, количество этих регистров довольно ограничено, и поэтому необходимо выполнять сжатие данных, чтобы уместить переопределения в один регистр.

4.5. Полезные функции из существующих решений

В процессе ознакомления с существующими системами частиц была выявлена полезная функциональность — возможность отслеживать ориентацию частиц в пространстве при помощи отладочных осей. Иными словами, отладочные оси позволяют увидеть текущую ориентацию локальной координатной системы частицы.

На рисунке 12 сверху изображено меню настроек, а снизу изображены включенные оси на частицах. В качестве осей по умолчанию вы-

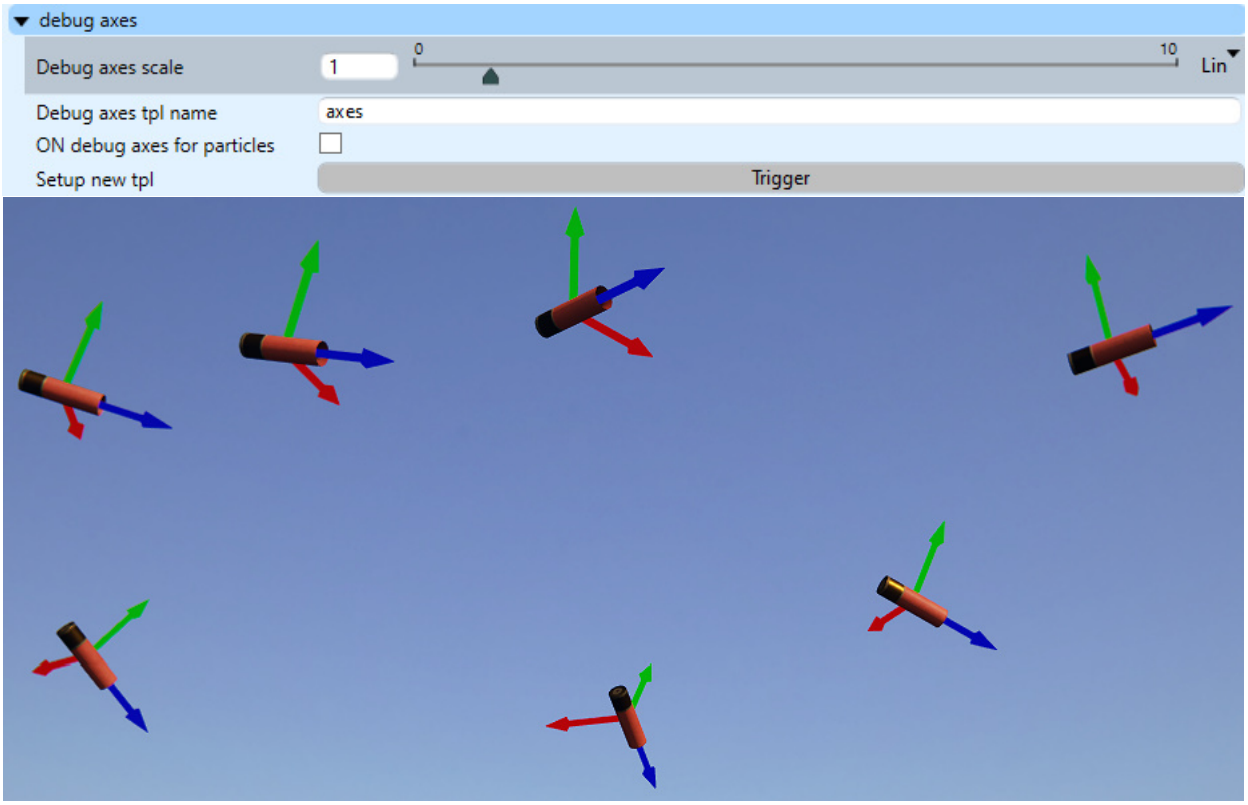

Рис. 12: Отладочные оси и их меню настроек

бран шаблон с названием “axes”. Однако предоставлена возможность выбрать любой другой шаблон, написав его имя в поле и нажав кнопку “Trigger”. Поскольку модели частицы и отладочных осей могут существенно различаться в размерах, была предоставлена возможность равномерно масштабировать модель отладочных осей.

В начале этапа симуляции выполняется загрузка шаблона, содержащего модель отладочных осей. Загрузка выполняется, если указанный шаблон не был загружен в систему ранее. Затем на этапе сбора данных для частиц будет выполняться построение их матриц трансформации. После построения матрицы вызывается функция, которая отладочным осям, представляющим текущую частицу, задает такую же матрицу трансформации. Однако диагональ матрицы отладочных осей умножается на указанный коэффициент масштаба из меню настроек. Иными словами, отладочные оси являются частицами, которые не были порождены эмиттером. Кроме того, информация об отладочных осях не учитывается в статистике частиц на сцене.

5. Апробация

Перед интеграцией в основную ветвь с исходным кодом движка в редакторе игровых уровней была выполнена предварительная апробация разработанной функциональности для системы геометрических частиц.

Для проверки корректности работы статистики частиц рассматривалось несколько случаев использования. На сцену было добавлено 5 эмиттеров, которые суммарно порождали порядка 7 тысяч частиц. Тестировалось своевременное и корректное обновление статистики при включении и отключении систем геометрических частиц на сцене, при добавлении новых систем и удалении существующих. Во всех рассмотренных сценариях статистика работала корректно.

Для проверки корректности работы переопределенных свойств материала на сцену в редакторе уровней был добавлен эмиттер с порожденными им частицами. В программе для настройки материалов и в редакторе уровней соответствующему свойству материала порожденных частиц задавалось одно и то же значение. После чего выполнялось сравнение двух визуальных результатов. Перед переопределением свойству в программе настройки материалов присваивалось значение, равное нулю. Одни и те же значения свойств материала в редакторе уровней и в программе настройки материалов давали одинаковый визуальный результат для всех восьми свойств.

Для проверки корректности работы отладочных осей рассматривалось несколько случаев использования. На сцену был добавлен эмиттер с порожденными им частицами. Выполнялось тестирование того, правильно ли загружается новая модель осей после порождения частиц. Проверялась загрузка модели осей с пустым полем названия или с несуществующим названием модели. В таком случае, выполнялась загрузка модели по умолчанию с названием “axes”. Таким образом, во всех рассмотренных сценариях отладочные оси работали корректно.

После прохождения ревью и интеграции в основную ветвь с исходным кодом движка новая функциональность была еще раз успешно апробирована программистами и арт-отделом компании.

Заключение

В ходе данной работы были получены следующие результаты.

- Изучена реализация системы геометрических частиц в движке Saber3D. В процессе исследования выявлены участки кода, нуждающиеся в рефакторинге.
- Проведен рефакторинг участков кода движка Saber3D, ответственных за покадровую обработку частиц, что повысило сопровождаемость текущей системы геометрических частиц движка Saber3D.
- Реализована следующая функциональность для системы геометрических частиц движка Saber3D:
  - неравномерное изменение масштаба частиц по трем осям;
  - изменение ориентации модели частицы во время полета в соответствии с текущим вектором направления;
  - окно, показывающее текущую статистику частиц на сцене;
  - переопределение нескольких SFX свойств материала частиц в редакторе уровней при помощи текстового поля или сплайна;
  - реализована дополнительная функциональность, взятая из существующих аналогов, по добавлению отладочных осей с заданием их масштаба для отслеживания ориентации частиц.
- Разработанный код прошел ревью инженерами-программистами графики компании ООО “Сабер Интерактив СГС” и был интегрирован в основную ветвь исходного кода движка Saber3D. Добавленная функциональность успешно прошла апробацию программистами и арт-отделом компании.

Список литературы

[1] Akenine-Möller Tomas, Haines Eric, Hoffman Naty. Real-Time Rendering, Fourth Edition. — P. 567–569.

[2] Amazon Lumberyard User Guide. — Access mode: `https://docs.aws.amazon.com/lumberyard/latest/userguide/lumberyard-intro.html` (online; accessed: 25.12.2020).

[3] Angela Gross. ChilliSource Game Engine Particle System Study // Graduate Student Theses, Dissertations, Professional Papers. — 2016. — Access mode: `https://scholarworks.umt.edu/etd/10813` (online; accessed: 15.12.2020).

[4] Hughes John F., Dam Andries Van, McGuire Morgan. Computer Graphics: Principles and Practice, Third Edition. — P. 350–351.

[5] Lander Jeff. GRAPHIC CONTENT: The ocean spray in your face. — 1998. — P. 13–19.

[6] Particle System Overview // Valve Developer Wiki. — Access mode: `https://developer.valvesoftware.com/wiki/Particle_System_Overview` (online; accessed: 25.12.2020).

[7] Particles Attributes Reference. — Access mode: `https://docs.aws.amazon.com/lumberyard/latest/userguide/particle-editor-reference.html` (online; accessed: 25.12.2020).

[8] Raappana Joona. Great Particles and How to Make Them // Development of guidelines for the Unity Particle System. — 2018.

[9] Reeves W. Particle Systems — a Technique for Modeling a Class of Fuzzy Objects // ACM Trans. Graph. — 1983. — Vol. 2. — P. 91–108.

[10] Saber Interactive, Company Overview. — Access mode: `https://saber3d.com/company/` (online; accessed: 25.12.2020).

[11] Source Engine // Valve Developer Wiki. — Access mode: https://developer.valvesoftware.com/wiki/Source (online; accessed: 25.12.2020).

[12] Sylvan Sebastian. Particle System Simulation and Rendering on the Xbox 360 GPU. — 2007.

[13] Tran Tommy. Unreal Engine 4 Particle Systems Tutorial. — 2017. — Access mode: https://www.raywenderlich.com/270-unreal-engine-4-particle-systems-tutorial (online; accessed: 26.12.2020).

[14] Walker Trey. Will Rock interview // GameSpot. — 2002. — Access mode: https://www.gamespot.com/articles/will-rock-interview/1100-2837358/ (online; accessed: 25.12.2020).