Содержание

Введение

С развитием компьютерных технологий компьютерная графика приобрела совершенно новый статус, поэтому сегодня она является основной технологией в цифровой фотографии, кино, видеоиграх, а также во многих специализированных приложениях. Было разработано большое количество алгоритмов отображения. Главными критериями, которые к ним предъявляются, являются реалистичность изображения и скорость отрисовки. Однако зачастую чем выше реалистичность, тем больше времени и памяти требуется для работы алгоритма.

Одной из тем моделирования является моделирование движения частиц. Имеется огромная потребность в качественной и эффективной отрисовке распространения частиц вируса. Особенно эта тема стала актуальной после начала пандемии короновируса. Пандемия COVID-19 повлияла на жизнь миллионов людей по всему миру. Помимо серьезных последствий для здоровья, пандемия также изменила нашу повседневную жизнь, перевернула рынок вакансий и подорвала экономическую стабильность. В данном курсовом проекте речь пойдет о моделировании распростарнения частиц вирусной инфекции.

Цель работы – разработать программное обеспечение для моделирования распространения частиц короновирусной инфекции в помещении:

- проанализировать методы и алгоритмы, моделирующие броуновское движение частиц;
- определить алгоритм, который наиболее эффективно справляется с поставленной задачей.

1 Аналитическая часть

1.1 Понятие броуновского движения

Броуновское движение (иногда называют Брауновское движение) – беспорядочное движение малых частиц, взвешенных в жидкости или газе, происходящее под действием молекул окружающей среды. Исследовано в 1827 г. Броуном (Браун; Brown), который наблюдал в микроскоп движение цветочной пыльцы, взвешенной в воде.

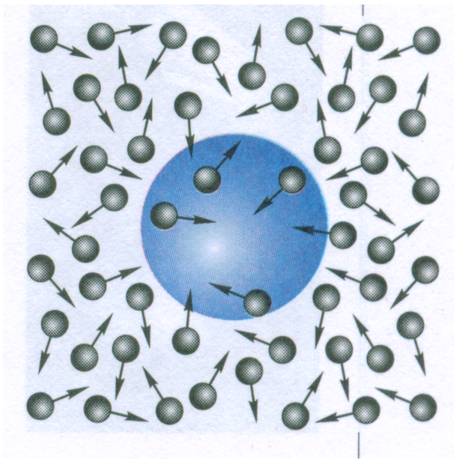

Рисунок 1.1 – Броуновское движение

Частицы размером около 1 мкм и менее совершают неупорядоченные независимые движения, описывая сложные зигзагообразные траектории. Интенсивность броуновского движения не зависит от времени, но возрастает с увеличением температуры, уменьшением вязкости и размеров частиц (независимо от их химической природы.)

Теория броуновского движения была построена независимо друг от друга Эйнштейном и Смолуховским в 1905-1906 гг. Причиной броуновского движения является тепловое движение молекул среды, проявляющееся в некомпенсированных ударах молекул о частицу, т.е. в флуктуациях давления. Эти удары приводят частицу в беспорядочное движение. Если отмечать положения частицы через равные небольшие промежутки времени, то траектория окажется сложной и запутанной.

Как показывают опытные данные, квадрат смещения частицы из начального положения в проекции на любую ось $\langle x^2 \rangle$ за время наблюдения τ, в отсутствие внешних сил определяется выражением $\langle x^2 \rangle = 2\mathrm{D}\tau$, где коэффициент диффузии броуновской (сферической) частицы $\mathrm{D} = \frac{kT}{6\pi\eta a}$, a – радиус частицы, η - коэффициент вязкости.

При описании броуновского движения частицы в одномерном случае можно считать, что на частицу действует сила случайная сила, среднее значение которой равно нулю $\langle F_x \rangle = \lim\limits_{t\to\infty}\{\frac{1}{t}\int_0^t F_x \mathrm{d}t\} = 0$ и сила сопротивления $F_c = \mathrm{r}v_x$, где r - коэффициент вязкого трения броуновской частицы в жидкости. Уравнение движения $ma_x = F_x$ - F_c при подстановке выражение для силы примет вид

$$m\ddot{x} + r\dot{x} = F_x \tag{1.1}$$

Умножим это урвнение на x и используем равенство $x\ddot{x} = \frac{\mathrm{d}(x\dot{x})}{\mathrm{d}t}$ - $\dot{x}^2$

$$m\frac{\mathrm{d}(x\dot{x})}{\mathrm{d}t} - m\dot{x}^2 + rx\dot{x} = xF_x \tag{1.2}$$

Проведем усреднение по времени

$$m\langle\frac{\mathrm{d}(x\dot{x})}{\mathrm{d}t}\rangle - m\langle\dot{x}^2\rangle + r\langle x\dot{x}\rangle = \langle xF_x\rangle \tag{1.3}$$

Тогда $\langle$ xF_x $\rangle = \lim\limits_{t\to\infty}\{\frac{1}{t}\int_0^t xF_x\mathrm{d}t\} = \lim\limits_{t\to\infty}\{x\frac{1}{t}\int_0^t F_x\mathrm{d}t - \frac{1}{t}\int_0^t(\int_0^t F_x\mathrm{d}t)\dot{x}\mathrm{d}t\} = 0$. Для одномерного движения по теореме о распределении энергии по степеням свободы $\frac{m\langle\dot{x}^2\rangle}{2} = \frac{kT}{2}$

Заменяем $\langle$ $\frac{\mathrm{d}(x\dot{x})}{\mathrm{d}t}$ $\rangle = \frac{\mathrm{d}\langle x\dot{x}\rangle}{\mathrm{d}t}$ и получаем уравнение m$\frac{\mathrm{d}\langle x\dot{x}\rangle}{\mathrm{d}t}$ + r$\langle$ x$\dot{x}$ $\rangle$ = kT, откуда

$$\langle x\dot{x}\rangle = \frac{kT}{r}(1 - e^{-\frac{m}{r}t}) \tag{1.4}$$

Для установившегося движения $\langle$ $x\dot{x}$ $\rangle = \frac{kT}{r}$. Так как x$\dot{x} = \frac{1}{2}$ $\frac{\mathrm{d}(x^2)}{\mathrm{d}t}$, то $\frac{\mathrm{d}\langle x^2\rangle}{\mathrm{d}t}$ $= \frac{kT}{r}$. После интеграрирования по времени получаем $\langle$ x^2 $\rangle = 2\frac{kT}{r}t$. Для сферической броуновской частицы, радиус которой равен a: $r = 6\pi\eta a$, поэтому $\mathrm{D} = \frac{kT}{6\pi\eta a}$.

Полученные выше формулы были экспериментально проверены в 1908 году Перреном, который измерял с помощью микроскопа перемещения броуновских частиц за одинаковые промежутки времени. Ему удалось на основании своих опытов с помощью этих формул определить постоянную Больцмана k и вычислить значение постоянной Авогадро N_A, совпадающие по величине с их значениями, полученными другими методами.

1.2 Моделирование броуновского движения

1.2.1 Классическое броуновское движение

Рассмотрим случайный процесс (случайную величину) *X(t)*, заданную на отрезке $[0, T]$.

Случайный процесс X(t) называется одномерным броуновским движением (или винеровским процессом) на интервале $[0, T]$, если он обладает следующими свойствами:

- $X(0) = 0$ почти наверное и *X(t)* - почти наверное непрерывная функция на $[0, T]$
- *X(t)* - процесс с независимыми приращениями
- *X(t)* - процесс с приращениями, распределёнными нормально.

Отметим следующие свойства броуновского движения:

- *X(t)* почти наверное нигде не дифференцируем
- *X(t)* - марковский процесс (не обладает памятью), т.е. если известна величина *X(t)*, то при $t_1 < t < t_2$ величины $X(t_1)$ и $X(t_2)$ независимы.
- Фрактальная размерность графика *X(t)* равна 1.5
- Приращение *X(t)* обладает свойством статистического самоподобия: для любого $r > 0$

$$X(t + \triangle t) = \frac{1}{\sqrt{r}}(X(t + r \triangle t) - X(t)) \tag{1.5}$$

- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени

$$D(X(t_2) - X(t_1)) = \sigma^2 |t_2 - t_1| \tag{1.6}$$

- Математическое ожидание приращения равно

$$E(|X(t_2) - X(t_1)|) = \sqrt{\frac{2}{\pi}}\sigma\sqrt{|t_2 - t_1|} \tag{1.7}$$

Для моделрования броуновского движения можно воспользоваться разными алгоритмами. Рассмотрим 3 из них.

Проще всего реализовать дискретную реализацию броуновского движения, рассмотрев последовательность $x_0 = 0$, $x_{n+1} = x_n + g_n$, где g_n - случайная величина, имеющая нормальное распределение (например, $N(0, 1)$).

$array[N]$
$array[0] \leftarrow 0$
for i = 1,..., N **do**
    $array[i+1] \leftarrow array[i] + randomNormal(0, 1)$
end for

1.2.2 Алгоритм срединных смещений

Метод случайного срединного смещения основан на работах Н.Виннера , он более сложен, чем метод из предыдущего параграфа, однако используется для конструктивного доказательства существования броуновского движения, а также для построения фрактальной интерполяции (когда необходимо чтобы кривая проходила через заданные точки интерполяции). Метод также может быть обобщен на случай n-мерных броуновских движений.

Алгоритм случайного срединного смещения вычисляет значения $X(t)$ в диадических рациональных точках вида $\frac{k}{2^n} \in [0,1]$. Последовательно вычисляются значения в середине отрезка [0,1], а затем в серединах отрезков $[0, \frac{1}{2}]$ и $[\frac{1}{2}, 1]$ и т.д. На каждом шаге итерации должен выполнятьяс закон дисперсии для приращения (1.9) в вычисленных точках. Параметр σ определяет масщтаб по вертикальной оси, не влияя на фрактальную размерность графика.

Броуновское движение методом срединнго смещения (1)

Вход: N, σ // N - число шагов алгоритма, при этом всего $2^N + 1$ точек интерполяции, σ - параметр вертикального масштаба, коэффициент дисперсии

Выход: массив значений $\left\{X(\frac{k}{2^N})\right\}_{k=0}^{2^N}$ // реализация броуновского движения $X(t)$ на дискретном множестве точек вида $t_k = \frac{k}{2^N}$, k $\in [0, 2^N]$

$X(0) \leftarrow 0$

$X(1) \leftarrow \sigma g$ // g - случайная величина, распределенная нормально с параметрами $N(0,1)$

$X(\frac{1}{2}) \leftarrow \frac{1}{2}(\mathrm{X}(0) + \mathrm{X}(1)) + \frac{1}{2}\sigma \mathrm{g}$

$X(\frac{1}{4}) \leftarrow \frac{1}{2}(\mathrm{X}(0) + \mathrm{X}(\frac{1}{2})) + \frac{1}{2^{\frac{3}{2}}}\sigma \mathrm{g}$

$X(\frac{3}{4}) \leftarrow \frac{1}{2}(\mathrm{X}(\frac{1}{2}) + \mathrm{X}(1)) + \frac{1}{2^{\frac{3}{2}}}\sigma \mathrm{g}$

...

$X(\frac{1}{2^N}) \leftarrow \frac{1}{2}\ (\mathrm{X}(0) + \mathrm{X}(\frac{1}{2^{N-1}})) + \frac{1}{2^{(N+1)/2}}\sigma \mathrm{g}$

...

$X(\frac{2^N-1}{2^N}) \leftarrow \frac{1}{2}\ (\mathrm{X}(\frac{2^{N-1}-1}{2^{N-1}}) + \mathrm{X}(1)) + \frac{1}{2^{(N+1)/2}}\sigma \mathrm{g}$

Заметим, что точки t_k можно последовательно занумеровать номерами k. При этом если точка имеет вид $\frac{a}{2^b}$, то ее номер $k = a2^{N-b}$. Укажем алгорит, в котором точки t_k пронумерованы эфективно.

Броуновское движение методом срединнго смещения (2)

Вход: N, σ // N - число шагов алгоритма, при этом всего $2^N + 1$ точек интерполяции, σ - параметр вертикального масштаба, коэффициент дисперсии

Выход: массив значений $\left\{X(\frac{k}{2^N})\right\}_{k=0}^{2^N}$ // реализация броуновского движения $X(t)$ на дискретном множестве точек вида $t_k = \frac{k}{2^N}$, k $\in [0, 2^N]$

$X(0) \leftarrow 0$

$X(1) \leftarrow \sigma g$ // g - случайная величина, распределенная нормально с параметрами $N(0,1)$

for j = 1,..., N **do**

    for i = 1,..., 2^{N-1} **do**

        $X((2i-1)2^{N-j}) \leftarrow X((i-1)2^{N-j+1}) + X(i2^{N-j+1}) + \frac{1}{2^{(j+1)/2}}\sigma$ g

    end for

end for

1.2.3 Фрактальное броуновское движение

Фрактальное броуновское движение (ФБД) уже не является марковским процессом, а обладает некорой "памятью". Кроме того, вводя параметр $0 < H < 1$ можно получитьодномерное ФБД размерности $d = 2 - H$ и двумерное ФБД размерности $d = 3 - H$. Заметим, что классическое броуновское движение получается как частный случай при $H = 0.5$. Для апроксимации ФБД нет простого метода, вроде суммирования нормальных случайных величин, как в случае классического броуновского движения. Для апроксимации ФБД наиболее удобно использовать преобразования Фурье.

Рассмотрим случайный процесс (случайную величину) $X(t)$, заданную на отрезке $[0, T]$.

Случайный процесс $X(t)$ называется одномерным фрактальным броуновским движением на интервале $[0, T]$, если он обладает следующими свойствами:

- $X(0) = 0$ почти наверное и $X(t)$ - почти наверне непрерывная функция на $[0, T]$
- $X(t)$ - процесс с приращениями, распределенными нормально

Отметим следующие свойства фрактального броуновского движения:

- $X(t)$ почти наверное нигде не дифференцируем
- Фрактальная размерность графика $X(t)$ равна $2 - H$
- Процесс $x(t)$ не обладает свойством независимости приращений
- Приращение *X(t)* обладает свойством статистического самоподобия: для любого $r > 0$

$$X(t + \triangle t) = \frac{1}{\sqrt{r}}(X(t + r \triangle t) - X(t)) \tag{1.8}$$

- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени

$$D(X(t_2) - X(t_1)) = \sigma^2 |t_2 - t_1|^{2H} \tag{1.9}$$

- Математическое ожидание приращения равно

$$E(|X(t_2) - X(t_1)|) = \sqrt{\frac{2}{\pi}}\sigma|t_2 - t_1|^H \tag{1.10}$$

Метод Фурье-фильтрации для построения ФБД

Теорема 1. *Если $X(t)$ - ФБД с параметром H, то его спектральная плотность*

$$S(f) \propto \frac{1}{f^{2H+1}} \tag{1.11}$$

Идея метода состоит в следующем. Строится преобразование Фурье для искомого ФБД в частной области, задавая случайные фазы и подбирая амплитуды, удовлетворяющие свойству из Теоремы 1. Затем получаем ФБД во временной области с помощью обратного преобразования Фурье.

Будем моделировать дискретный аналог ФБД, то есть наша цель- получить величины $\{X_n\}_{n=0}^{N-1}$, апроксимирующиеФБД в точках n. Воспользуемся формулой дискретного преобразования Фурье

$$\hat{X}_n = \sum_{k=0}^{N-1} X_k e^{-2\pi kn/N} \tag{1.12}$$

и обратного дискретного преобразования Фурье

$$X_n = \sum_{k=0}^{N-1} \hat{X}_k e^{2\pi kn/N} \tag{1.13}$$

Далее будем рассматривать только четные значения N, а для применения метода *быстрого дискретного преобразования Фурье* нужно, чтобы $N = 2^M$, $M \in \mathbb{N}$. Метод быстрого дискретного преобразования Фурье реализован во многих системах компьютерной алгебры. Он позволяет сократить вычисления в $\frac{2N}{\log_2 N}$ раз.

Для того, чтобы получающиеся величины X_n были вещественными мы наложим условие сопряженной симметрии:

$$\hat{X}_0, \hat{X}_{N/2} \in \mathbb{R}, \hat{X}_n = \hat{X}_{N-n}, n = 1, ..., N/2 - 1 \tag{1.14}$$

Фильтрация относится к той части моделирования, когда мы заставляем коэффициенты преобразования Фурье удовлетворять степенному закону из Теоремы 1:

$$|\hat{X}_n|^2 \propto \frac{1}{n^{2H+1}}, n = 1, ..., N/2 \tag{1.15}$$

Для этого возьмем

$$\hat{X}_n = \frac{ge^{2\pi iu}}{n^{H+0.5}} \tag{1.16}$$

где g - независимые значения нормально распределенной случайной величины с параметрами $N(0,1)$, а u - независимые значения равномерно распределенной на отрезке [0,1] случайной величины. Оставшиеся коэффициенты вечислим из сотношений 1.15.

Для вычисления искомой аппроксимации ФБД $\{X_n\}_{n=0}^{N-1}$ применим обратное дискретное преобразование Фурье к набору $\left\{\hat{X}_n\right\}_{n=0}^{N-1}$.

Кривая ФБД методом Фурье-фильтрации

Вход: $H \in (0,1)$, $N = 2^M$, $M \in \mathbb{N}$ // H - параметр ФБД, размерность графика равна $d = 2 - H$, N - параметр, определяющий количество точек дискретизации ФБД.

Выход: массив значений $\{X_n\}_{n=0}^{N-1}$ // дискретная апроксимация ФБД в последовательные моменты времени n.

$\hat{X}_0 \leftarrow g$
for j = 1,..., N/2-1 **do**
    $\hat{X}_j \leftarrow \frac{ge^{2\pi iu}}{j^{H+0.5}}$
end for
$\hat{X}_{N/2} \leftarrow \frac{g\cos(2\pi iu)}{(N/2)^{H+0.5}}$ // Здесь cos — вещественная часть комплексной экспоненты e
for j = N/2+1,..., N-1 **do**
    $\hat{X}_j \leftarrow \overline{\hat{X}_{N-j}}$
end for
$X \leftarrow convert(\hat{X})$ // Вектор $X = \{X_0, ..., X_{N-1}\}$ получается обратным дискретным преобразованием Фурье из вектора $\hat{X} = \left\{\hat{X}_0, ..., \hat{X}_{N-1}\right\}$.

Для построения апроксимации двумерного фрактального броуновского движения методом Фурье-фильтрации используются те же идеи, что и в одномерном случае. Вместо $\hat{X}_n$ используется $\hat{X}_{k,j}$, $k, j = \overline{0, N-1}$, условие Теоремы 1 примет вид:

$$|\hat{X}_{k,j}|^2 \propto \frac{1}{(k^2+j^2)^{H+1}}, n, k = 1, ..., N/2 \tag{1.17}$$

мы возьмем

$$\hat{X}_{k,j} = \frac{ge^{2\pi iu}}{(k^2+j^2)^{H/2+0.5}}, n, k = 1, ..., N/2 \tag{1.18}$$

Запишем обратное дискретное преобразование Фурье: для $m, n = \overline{0, N-1}$

$$\hat{X}_{m,n} = \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} \hat{X}_{k,j} e^{-2\pi i \frac{kn+jm}{N}} = \hat{X}_{0,0} + \sum_{k=1}^{N-1} \hat{X}_{k,0} e^{-2\pi i \frac{kn}{N}} + \sum_{j+1}^{N-1} \hat{X}_{0,j} e^{-2\pi i \frac{jm}{N}} + \\ \sum_{k=1}^{N/2}\sum_{j=1}^{N/2} \hat{X}_{k,j} e^{-2\pi i \frac{kn+jm}{N}} + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=\frac{N}{2}+1}^{N-1} (...) + \sum_{k=1}^{N/2}\sum_{j=\frac{N}{2}+1}^{N-1} (...) + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=1}^{N/2} (...) \tag{1.19}$$

Из формулы (1.19) следует, что для вещественности всех величин $X_{m,n}$ достаточно выполнения следующих условий сопряженной симметрии:

$$\hat{X}_{N-k,N-j} = \overline{\hat{X}_{k,j}} \qquad k, j = \overline{1, N/2} \qquad \hat{X}_{N/2,N/2} \in \mathbb{R} \tag{1.20}$$

$$\hat{X}_{k,N-j} = \overline{\hat{X}_{N-k,j}} \qquad k, j = \overline{1, N/2-1} \qquad \hat{X}_{0,0} \in \mathbb{R} \tag{1.21}$$

$$\hat{X}_{0,N-j} = \overline{\hat{X}_{0,j}} \qquad j = \overline{1, N/2} \qquad \hat{X}_{0,N/2} \in \mathbb{R} \tag{1.22}$$

$$\hat{X}_{N-k,0} = \overline{\hat{X}_{k,0}} \qquad k = \overline{1, N/2} \qquad \hat{X}_{N/2,0} \in \mathbb{R} \tag{1.23}$$

Условия (1.22)-(1.23) обеспечивают вещественность первых двух сумм, а условия (1.20)-(1.21) - оставшихся четырех сумм.

Поверхность ФБД методом Фурье-фильтрации

Вход: $H \in (0,1)$, $N = 2^M$, $M \in \mathbb{N}$ // H - параметр ФБД, размерность графика равна $d = 3 - H$, N - параметр, определяющий количество точек

ФБД по каждому из двух измерений.

Выход: массив значений $\{X_{n,k}\}_{n,k=0}^{N-1}$ // дискретная апроксимация ФБД на решетке узлов.

for $j, k = 1, ..., N/2$ **do**
  $\hat{X}_{j,k} \leftarrow \frac{ge^{2\pi iu}}{(j^2+k^2)^{H/2+0.5}}$
  $\hat{X}_{N-j,N-k} \leftarrow \overline{\hat{X}_{j,k}}$
end for
for $k = 1, ..., N/2 - 1$ **do**
  $\hat{X}_{0,k} \leftarrow \frac{ge^{2\pi iu}}{(k^2)^{H/2+0.5}}$
  $\hat{X}_{k,0} \leftarrow \frac{ge^{2\pi iu}}{(k^2)^{H/2+0.5}}$
  $\hat{X}_{0,N-k} \leftarrow \overline{\hat{X}_{0,k}}$
  $\hat{X}_{N-k,0} \leftarrow \overline{\hat{X}_{k,0}}$
end for
for $j, k = 1, ..., N/2 - 1$ **do**
  $\hat{X}_{N-j,k} \leftarrow \frac{ge^{2\pi iu}}{((N-j)^2+k^2)^{H/2+0.5}}$
  $\hat{X}_{j,N-k} \leftarrow \overline{\hat{X}_{N-j,k}}$
end for
$\hat{X}_{0,0} \leftarrow 0$
$\hat{X}_{N/2,0} \leftarrow \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$
$\hat{X}_{0,N/2} \leftarrow \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$
$\hat{X}_{N/2,N/2} \leftarrow \frac{g\cos(2\pi u)}{(2(N/2)^2)^{H/2+0.5}}$
$X \leftarrow convert(\hat{X})$ // Обратное дискретное преобразование Фурье матрицы $\hat{X} = \{X_{j,k}\}_{j,k=0}^{N-1}$.

Вывод

Наиболее реалистично изобразить броуновское движение позволяет алгоритм Фурье-фильтрации. Однако он содержит большое количество сложных вычеслений, которые отрицательно влияют на скорость работы программы. Поэтому для реализации отрисовки броуновского движения будет использован метод срединного смещения. Данный метод легко обобщается для случая n-мерных броуновских движений, а также требует более простых вычислений.

1.3 Формализация модели

Модель броуновского движения частиц будет задаваться такими характеристиками, как:

- размер частиц – число типа int;
- скорость распространения – маляа, средняя или высокая;
- количество частиц – число типа int.

Также частью на сцене будет изображено помещение. Пользователь должен уметь задавать материал покрытия стен:

- дерево;
- бумага (обои).

и пола:

- дерево (паркет);
- керамика (плитка).

1.4 Выбор программного обеспечения

Рендеринг или отрисовка – термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы.

В языке Java есть несколько основных инструментов для создания пользовательских изображений. Самыми популярными из них являются JavaFX и Swing.

1.4.1 Swing

Swing – библиотека для создания графического интерфейса для программ на языке Java. Swing был разработан компанией SunMicrosystems. Он содержит ряд графических компонентов, таких как кнопки, поля ввода, таблицы и т. д.

Преимущества:

- Кроссплатформенность;
- Компоненты Swing следуют парадигме Model-View-Controller (MVC) и, таким образом, могут обеспечить гораздо более гибкий пользовательский интерфейс;
- Swing обеспечивает встроенную двойную буферизацию.

Недостатки:

- достаточно узкий спектр возможностей при работе с ui.
- считается устаревшей библиотекой.

1.4.2 JavaFX

JavaFX – платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений, работающих в браузерах, и для приложений на мобильных устройствах.

JavaFX предназначен для предоставления приложениям таких сложных функций графического интерфейса, как плавная анимация, просмотр веб-страниц, воспроизведение аудио и видео, а также использование CSS стилей.

Преимущества:

- кроссплатформенность;
- больше встроенных возможностей;
- меньше кода.

Недостатки:

- технология еще молодая и "незрелая";

- в значительной степени зависит от огромной инфраструктуры, которая окружает Java.

Вывод

Уже более 10 лет разработчики приложений считают Swing высокоэффективным инструментарием для создания графических пользовательских интерфейсов (GUI) и добавления интерактивности в Java-приложения. Однако некоторые из самых популярных на сегодняшний день функций графического интерфейса не могут быть легко реализованы с помощью Swing в отличии от JavaFX.

Также можно писать программы на JavaFX, используя гораздо меньше кода, потому что JavaFX выполняет за нас всю работу. Не нужно регистрировать event listeners, и это делает тело функций более кратким. Кроме того, с помощью механизма привязки JavaFX легко интегрировать компоненты графического интерфейса с базовой моделью. Основываясь на вышесказанном в качестве библиотеки для работы с GUI была выбрана JavaFX.

Вывод

В данном разделе были формально описаны все методы по визуализации броуновского движения, с помощью которых можно получить реализацию распространения частиц, а также техноогии работы с GUI. В качестве алгоритма визуализации броуновского дижения предпочтение отдается методу срединных смещений. Также JavaFX была выбрана в качестве библиотеки работы с GUI.

2 Конструкторская часть

В данном разделе будут представлены требования к программному обеспечению, а также схемы алгоритмов, выбранных для решения задачи.

2.1 Требования к программному обеспечению

Программа должна предоставлять доступ к функционалу:

- изменение параметров модели водопада в активном режиме: высота водопада, угол падения воды, скорость водяного потока;
- изменение параметров частиц, из которых состоит водопад, в активном режиме: количество частиц, размер частиц;
- включение и выключение работы модели водопада;
- вращение, перемещение и масштабирование модели.

Требования, которые предъявляются к программе:

- время отклика программы должно быть менее 1 секунды для корректной работы в интерактивном режиме;
- программа должна корректно реагировать на любое действие пользователя.

2.2 Разработка алгоритмов

В данном разделе будут представлены схемы реализации выбранных алгоритмов.

2.2.1 Система частиц для реализации водопада

На рисунке 2.1 показана схема алгоритма реализации движения частицы и возможные этапы ее превращения в пар и брызг.

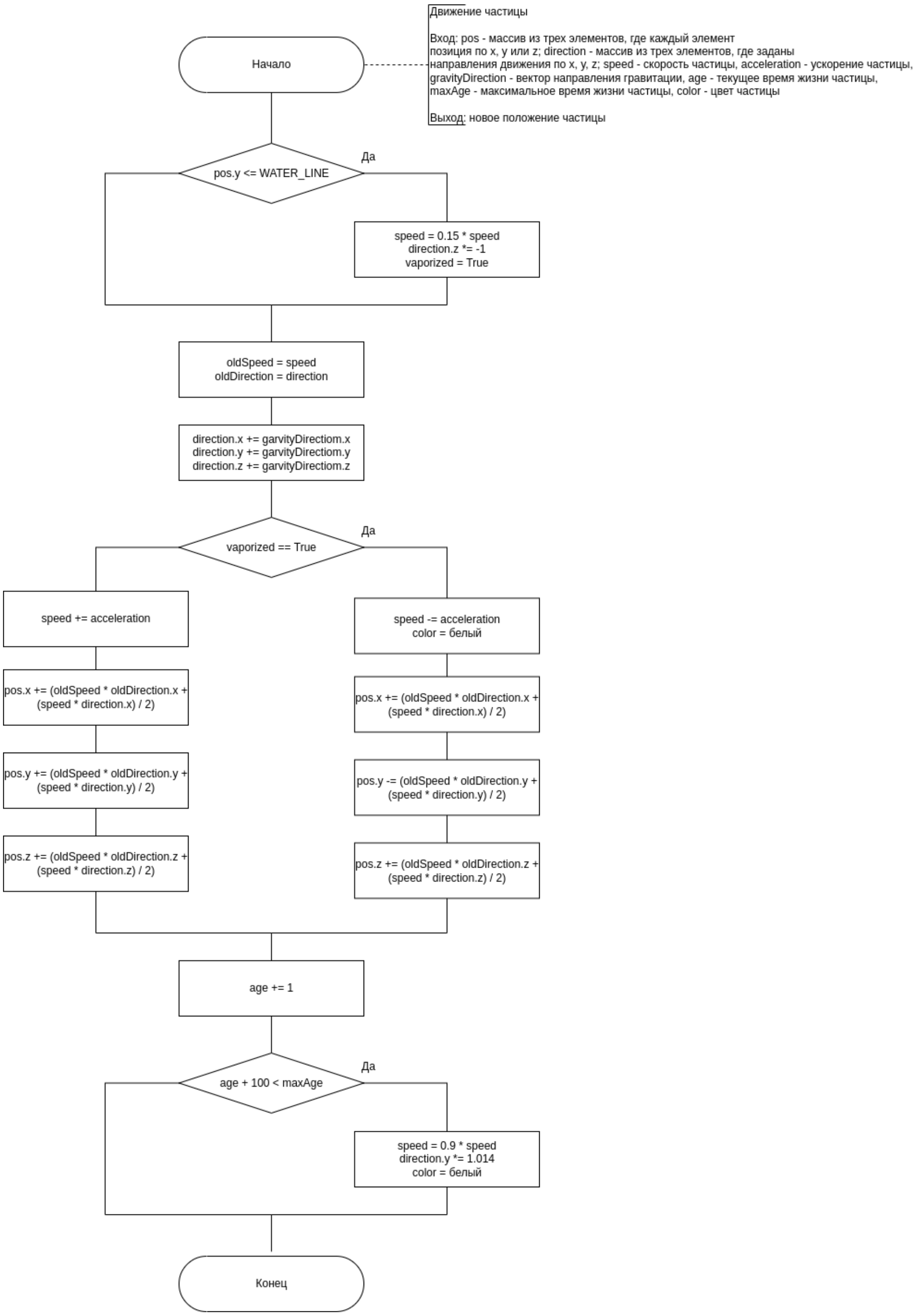

Рисунок 2.1 – Схема алгоритма движения частицы водопада

2.2.2 Отрисовка изображения

На рисунке 2.2 показана схема алгоритма отрисовки части сцены, отвечающей за водопад, а на рисунке 2.3 – схема алгоритма для отрисовки остальных объектов сцены (скалы и водного полотна).

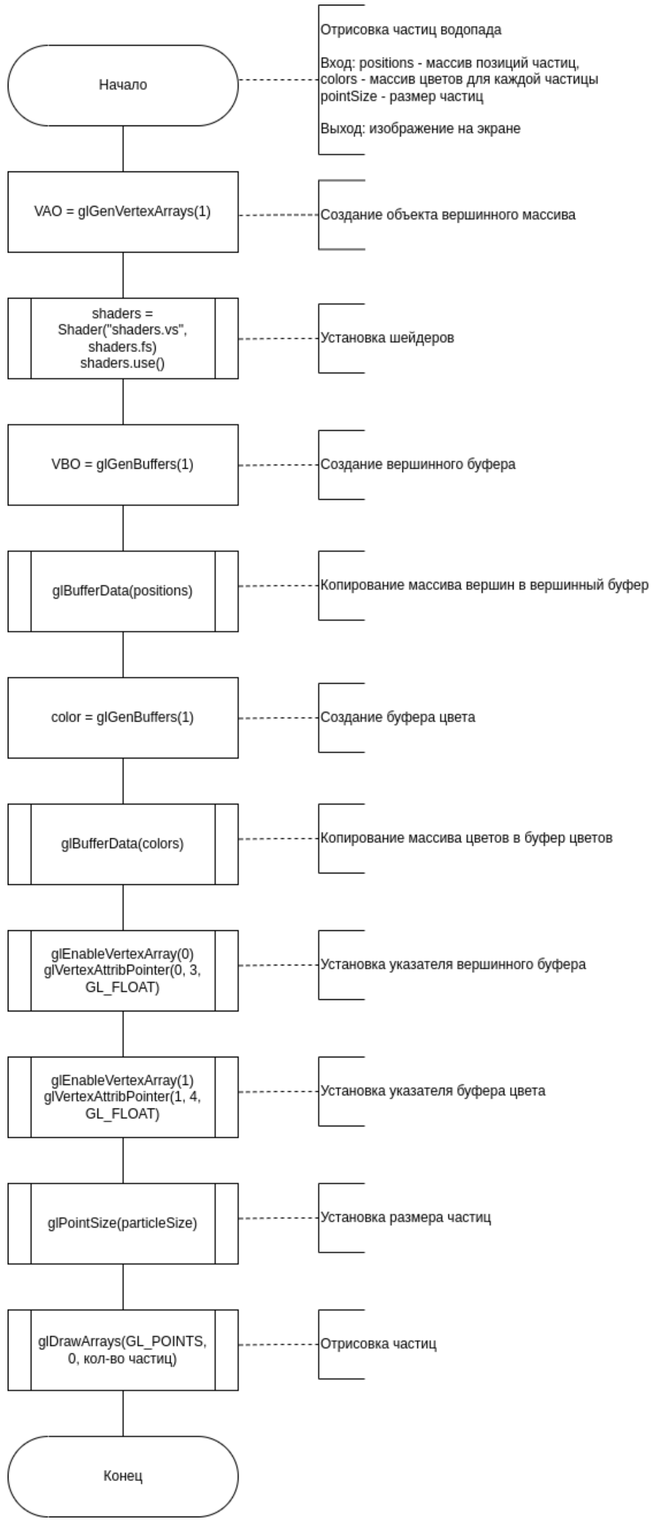

Рисунок 2.2 – Схема алгоритма отрисовки водопада

Рисунок 2.3 – Схема алгоритма отрисовки статичного объекта (скалы)

2.3 Описание структуры программы

На рисунке 2.4 показана структура реализуемых классов.

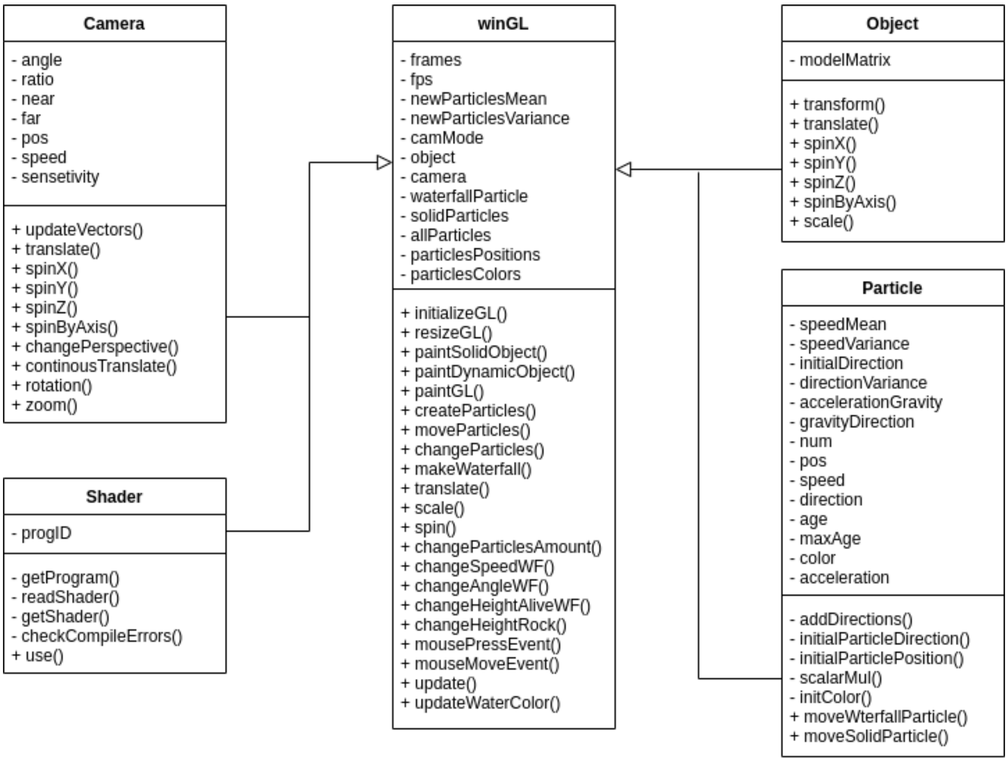

Рисунок 2.4 – Схема классов программы

Описание реализуемых классов:

- **Camera** – класс для работы с камерой. Хранит позицию камеры, угол ее наклона, скорость перемещения для динамической смены положения камеры;
- **Shader** – класс, который подключает шейдеры к приложению. Хранит данные шейдеров для передачи цвета объектов;
- **Object** – класс, который владеет информацией о всех объектах сцены;
- **winGL** – класс для отрисовки объектов сцены. Хранит массив частиц для водопада, хранит массив цветов для водопада, а также данные для отрисовки статичных объектов;
- **Particle** – класс, который описывает одну частицу водопада. Хранит ее положение, направление движения, цвет, скорость, время жизни и максимальное время жизни.

2.4 Используемые типы и структуры данных

При реализации программного обеспечения были использованы следующие типы и структуры данных:

- параметры водопада: высота, ширина, скорость, размер частиц – числа типа $float$;
- параметры водопада: количество частиц – число типа int
- точка – массив координат по осям X, Y, Z;
- объект – массив точек с координатами вершин, также массив связей между номерами вершин;
- водопад – массив частиц (объектов класса $Particle$).

Вывод

В данном разделе были рассмотрены требования, которые выдвигаются программному продукту, схемы алгоритмов, а также типы и структуры данных, которые были использованы при реализации ПО.

3 Технологическая часть

В данном разделе будут рассмотрены средства разработки программного обеспечения, детали реализации, а также диаграмма классов.

3.1 Средства реализации

При написании программного продукта был выбран язык *Python* [?]. Это обусловлено следующими факторами:

- объектно-ориентированный язык, что позволяет использовать структуру классов;
- имеются необходимые библиотеки для реализации поставленной задачи;
- существует много учебной литературы.

В качестве разработки интерфейса был выбран *Qt Designer* [?]. Он позволяет создать качественный интерфейс, так как имеет встроенный редактор выводимого окна.

В качестве среды разработки был выбран *Visual Studio Code* [?]. Данное приложение имеет большое количество плагинов для работы с кодом.

3.2 Реализация алгоритмов

В листинге 3.1 представлен алгоритм перемещения частицы водопада за один кадр.

Листинг 3.1 – Алгоритм перемещения частицы водопада за один кадр

```
def moveWaterfallParticle(self):
    if (self.pos[1] <= WATER_LINE):
        self.speed = BOUNCE_COEF * self.speed
        self.direction[2] *= -1
        self.vaporized = True

    oldSpeed = self.speed
```

```
    oldDirection = deepcopy(self.direction)

    self.direction[0] += self.gravityDirection[0]
    self.direction[1] += self.gravityDirection[1]
    self.direction[2] += self.gravityDirection[2]

    if (self.vaporized):
        self.speed -= self.acceleration

        self.pos[0] = self.pos[0] + oldSpeed * oldDirection[0] + (self.speed * self.direction[0]) / 2
        self.pos[1] = self.pos[1] - (oldSpeed * oldDirection[1] + (self.speed * self.direction[1]) / 2)
        self.pos[2] = self.pos[2] - (oldSpeed * oldDirection[2] + (self.speed * self.direction[2]) / 2)
    else:
        self.speed += self.acceleration

        self.pos[0] = self.pos[0] + oldSpeed * oldDirection[0] + (self.speed * self.direction[0]) / 2
        self.pos[1] = self.pos[1] + oldSpeed * oldDirection[1] + (self.speed * self.direction[1]) / 2
        self.pos[2] = self.pos[2] + oldSpeed * oldDirection[2] + (self.speed * self.direction[2]) / 2

    self.age += 1

    if (self.age + 100 > self.maxAge):
        self.speed *= 0.9
        self.direction[2] *= 1.014
        self.color = glm.vec4(1, 1, 1, 1)

    return self
```

Вывод

В данном разделе были рассмотрены средства, с помощью которых было реализовано ПО, описана структура классов проекта, а также представлен листинг алгоритма перемещения частицы водопада за кадр.

Заключение

В ходе программного обеспечения было разработано программное обеспечение, которое реализует модель водопада по методу частиц. Полученное приложение позволяет изменять параметры водопада в интерактивном режиме (скорость течения воды, угол наклона падающей воды, высота водопада, размер частиц, количество частиц). В процессе выполнения данной работы были выполнены следующие задачи:

- рассмотрены методы реализации модели водопада;
- выбран алгоритм, который наиболее эффективно решает поставленную задачу;
- реализован выбранный алгоритм;
- разработана структура классов проекта;
- проведен эксперимент по замеру производительности полученного программного обеспечения.

В процессе исследования было выявлено, что производительность программы понижается экспоненциально при линейном увеличении количества частиц. При этом водопад визуально выглядит лучше при наибольшем количестве частиц, которые вместе составляют единую структуру.