переспективного преобразования, и самого этапа растеризации, где по известным вершинам (в данном случае уже пикселям на экране) необходимо получить все остальные пиксели модели и их атрибуты (цвет, текстура, глубина).

Будут реализованы два алгоритма: растеризация — это ускорит отрисовку сцены в режиме реального времени, и рэйкастинг — данный алгоритм будет так же использован в выборе элемента щелчком мыщи с тем отличием, что будет рассматриваться не весь экран, а только выбранный пиксель.

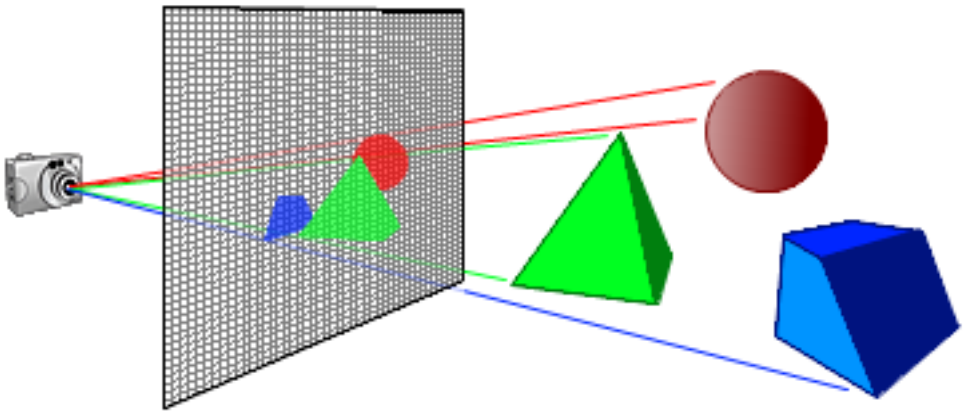

Рис. 1.1 – Процесс работы рэйкастинга

1.4 Описание трёхмерных преобразований

1.4.1 Способ хранения декартовых координат

Координаты можно хранить в форме вектор-столбца x, y, z. Однако в этом случае неудобно применять преобразования поворота, так как такой вектор нельзя умножить на соответствующие квадратные матрицы трансформации размерности четыре на четыре. Целесообразнее использовать вектор-столбцы размерности четыре - [x, y, z, w], где координата $w = 1$. Преобразования координат выполняются умножением слева преобразуемого вектора-строки на соответствующую матрицу линейного оператора.

таким образом, что чем дальше от наблюдателя находится вершина, тем больше становится это w-значение. После преобразования координат в пространство отсечения x, y попадают в диапазон от -w до w, а вершина z от 0 до w (вершины, находящиеся вне этого диапазона, отсекаются). Следующий этап — спроецировать все координаты на одну плоскость, разделив всё на координату z. После умножения вектора координат на матрицу перспективной проекции, реальная координата z заносится в w-компоненту, так что вместо деления на z делят на w.

$$\begin{pmatrix} \frac{\cot\frac{fov}{2}}{aspect} & 0 & 0 & 0 \\ 0 & \cot\frac{fov}{2} & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & 1 \\ 0 & 0 & \frac{-2fn}{f-n} & 0 \end{pmatrix}. \tag{1.7}$$

В (1.7):

- aspect — отношение ширины изображения к его высоте;
- fov — угол обзора камеры;
- f — координата z дальней от камеры плоскости отсечения пирамиды видимости;
- n — координата z ближней к камере плоскости отсечения пирамиды видимости

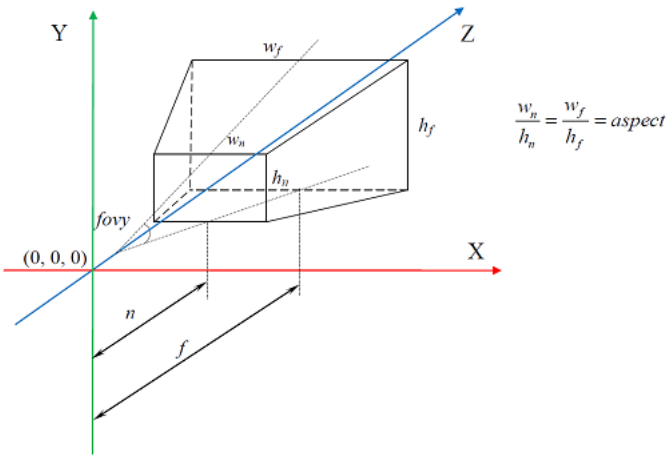

Рис. 1.2 – Усеченная пирамида проекции

1.4.6 Преобразования трехмерной сцены в пространство области изображения

Чтобы преобразовать спроецированные координаты в координаты области изображения, нужно инвертировать ось Y и применить операции переноса и маштабирования к каждой вершине.

1.5 Анализ алгоритма удаления невидимых ребер и поверхностей

1.5.1 Z-буфер

Суть данного алгоритма — это использование Z-буфера, в котором хранятся информация о координате Z для каждого пикселя. Первоначально в Z-буфере находятся максимально возможные значения Z. Каждый многоугольник преобразуется в растровую форму. В процессе подсчета глубины нового пикселя, он сравнивается с тем значением, которое уже лежит в Z-буфере. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то онотрисовывается и происходит корректировка Z-буфера. Необходимо отметить, что в данном алгоритме может возникнуть следующая ситуация: если два объекта имеют близкую Z-координату, то они могут перекывать друг друга. Это называется Z-конфликт (рис. 1.3). Решаются Z-конфликты сдвигом одного объекта относительно другого на величину, превышающую погрешность Z-буфера.

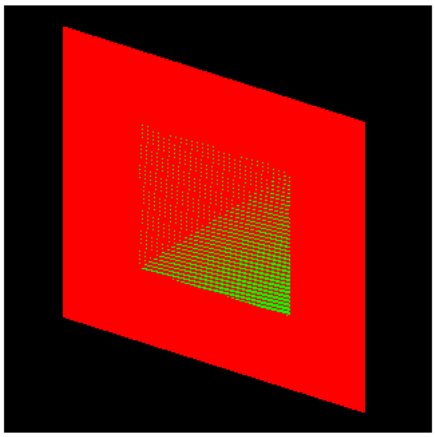

Рис. 1.3 – Z-конфликт

точки. Правой кнопкой мыши можно добавть новую точку и соответственно линию на модель. Для обзора сцены используется камера, управление которой осуществляется посредством нажатия клавиш на клавиатуре. При нажатии кнопки <выбор трансформирования> откроется окно (рис. 3.3), в котором можно задать необходимые матрицы аффиных преобразований.

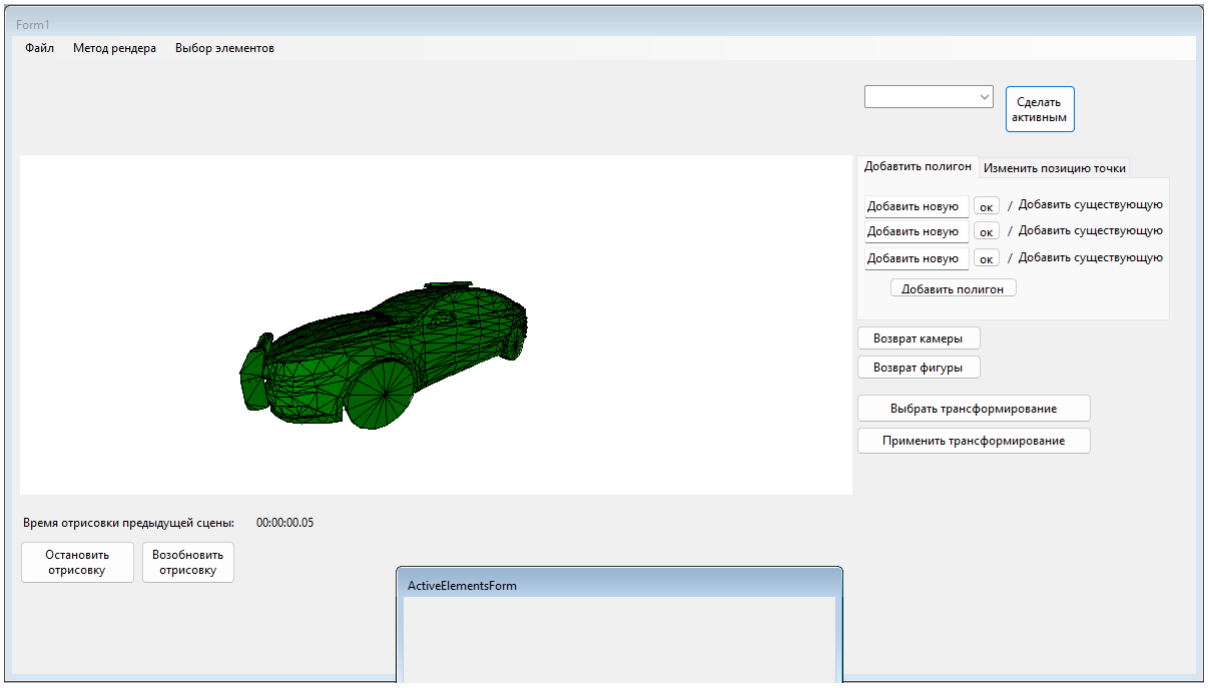

Рис. 3.2 – Интерфейс программы

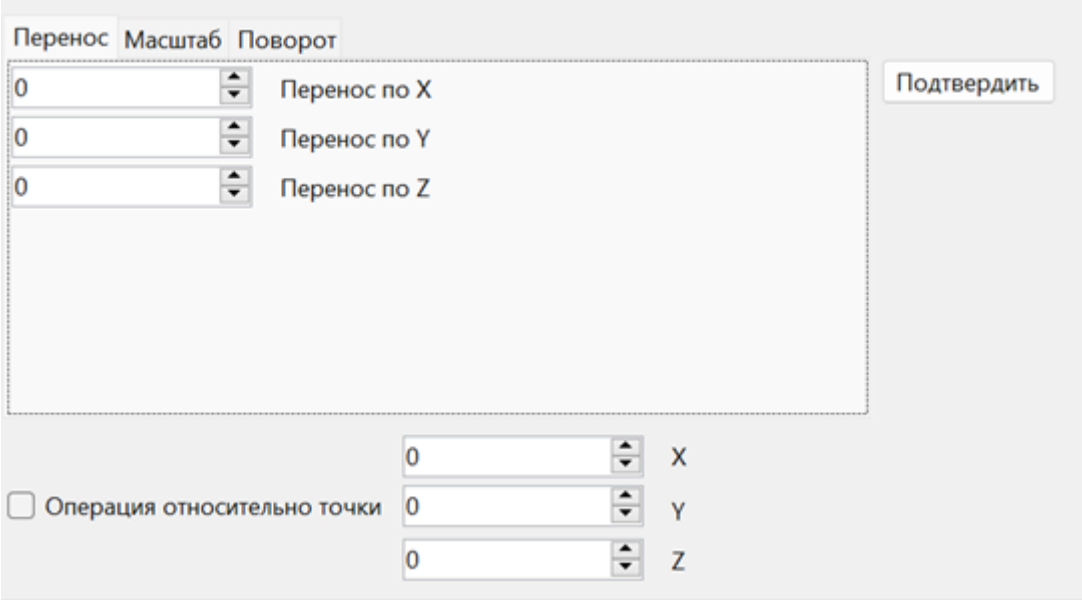

Рис. 3.3 – Интерфейс программы

4.4 Время выполнения реализаций алгоритмов

На рисунке 4.1, приведены графические результаты замеров времени работы алгоритма рейкастинга для параллельного (при количестве потоков, равном количеству логических ядер ЭВМ, на которой проводились замеры времени, затрачиваемого реализацией трассировки лучей) и последовательного случаев при разной доле заполнения экрана. На рисунке 4.2, приведены графические результаты замеров времени работы алгоритмов растеризации при разной доле заполнения экрана.

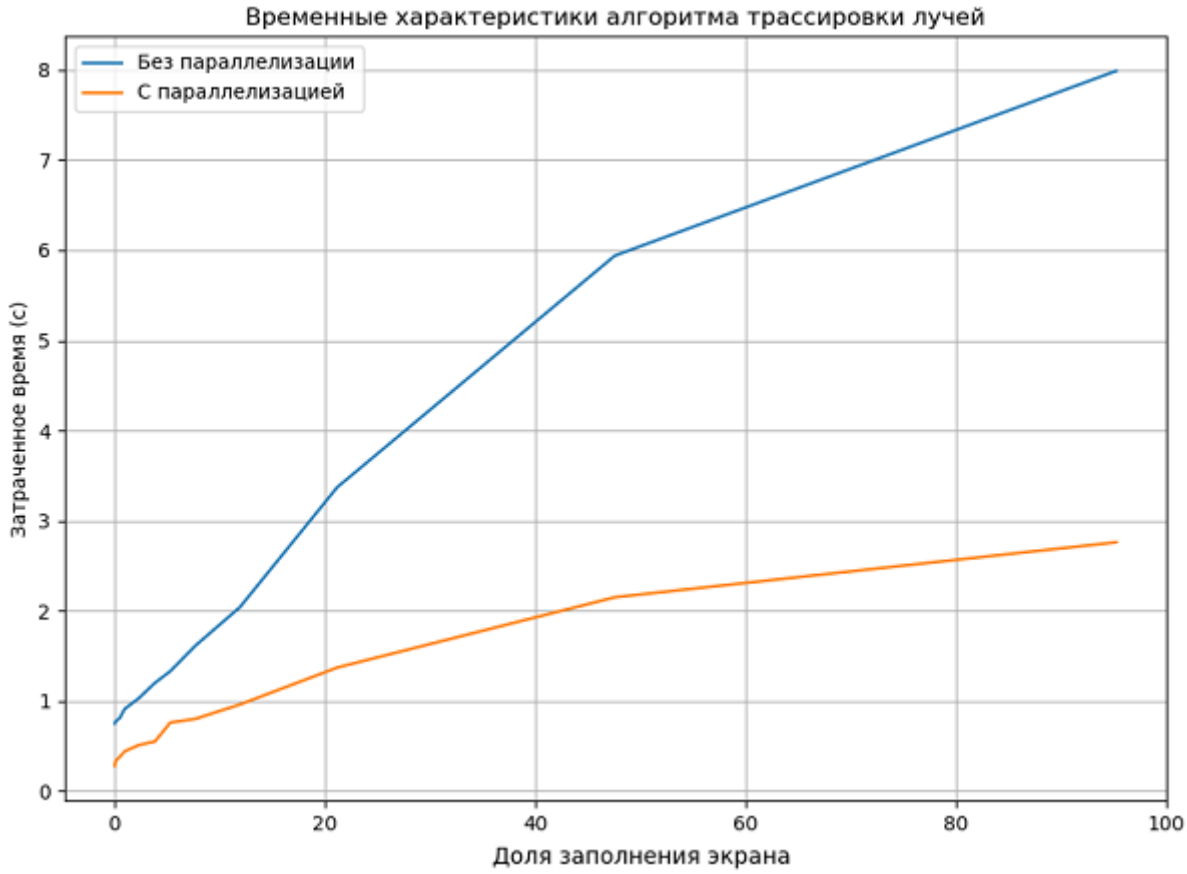

Рис. 4.1 – Время работы реализаций алгоритма

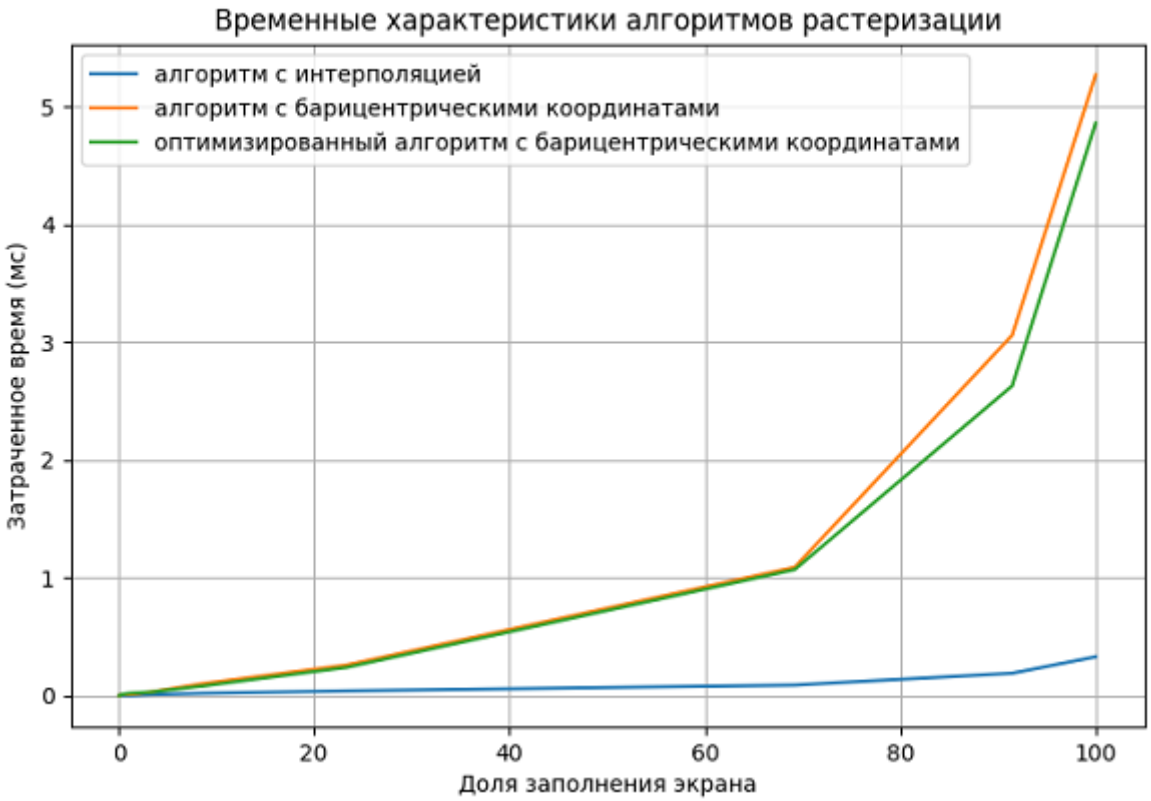

Рис. 4.2 – Время работы реализаций алгоритма

4.5 Примеры работы программы

На рисунках 4.3 и 4.4 приведены прменры работы программы для алгоритмов растеризации использующих интреполяцию и барицентрические координаты соответственно. На экран выводиться лицевая сторона куба.

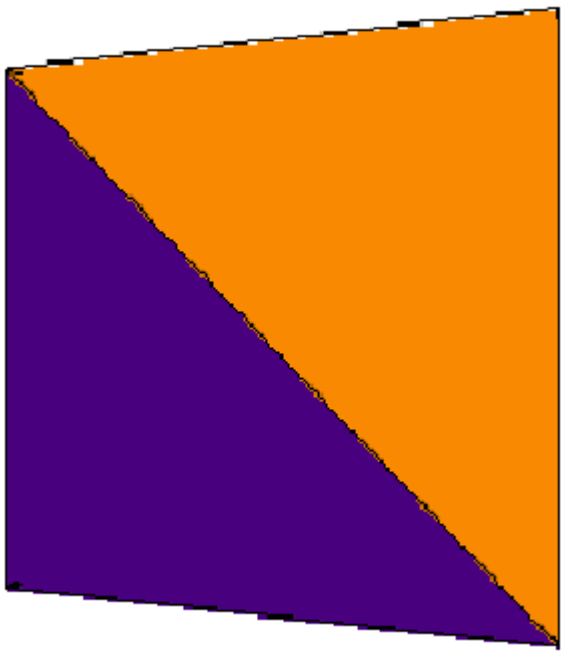

Рис. 4.3 – Пример работы программы

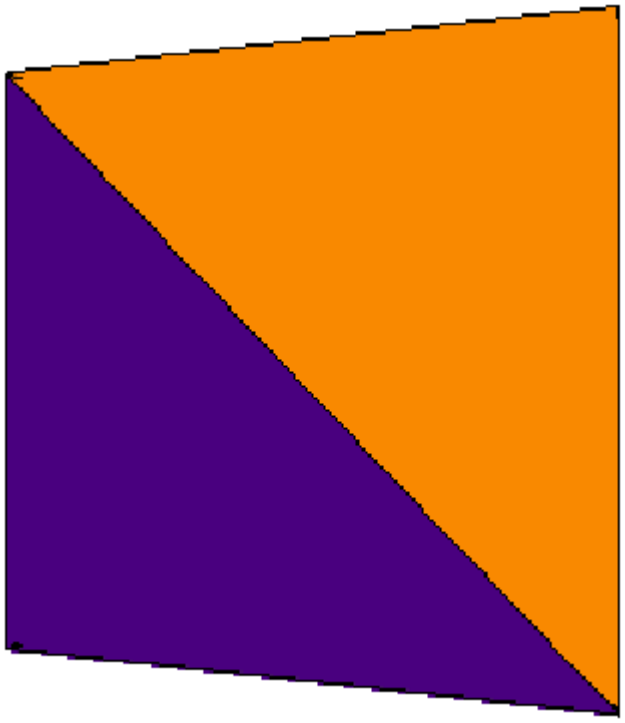

Рис. 4.4 – Пример работы программы