



Московский Государственный Университет им. М.В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Интеллектуальных Информационных Технологий

Майоров Николай Дмитриевич

Универсальные дифференцируемые представления объектов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

к.ф.-м.н.

В.А.Фролов

Москва, 2024

Аннотация

Данная работа посвящена созданию нескольких дифференцируемых процедурных генераторов. Каждый из них способен получать объект своего класса по параметрам и матрицу производных выходных элементов по входным данным. Полученные генераторы используются в задаче трёхмерной реконструкции. В работе также представлена идея универсального дифференцируемого процедурного генератора, основанного на деревьях. Он может создавать модели вне зависимости от их класса, и также возвращает производные по параметрам. Генератор легко дополняется новыми элементами и масштабируется при необходимости.

Содержание

1 Введение	4
2 Постановка задачи	5
2.1 Цели работы	5
2.2 Формальная постановка задачи	6
3 Обзор литературы	7
3.1 Представления объектов	7
3.2 Процедурная генерация	7
3.3 Реконструкция трёхмерных моделей	8
4 Предложенные методы	9
4.1 Генераторы посуды и зданий	9
4.1.1 Подсчёт производных	9
4.1.2 Процедурный генератор посуды	9
4.1.3 Процедурный генератор зданий	10
4.2 Универсальные генераторы	11
4.2.1 Структура универсального генератора	11
4.2.2 Виды узлов в дереве генератора	12
4.2.3 Подсчёт производных в узлах	13
4.2.4 Сложные узлы	16
4.2.5 Реализованный функционал	17
5 Процедурная генерация в задаче реконструкции	22
5.1 Генераторы посуды и зданий	22
5.2 Универсальный генератор	25
6 Заключение	26

1 Введение

Индустрія комп'ютерних ігор і кіно – це сектор економіки, пов'язаний з ним розвитком, продажами та розробкою. Зародившись просто як необичне хоббі групами зацікавлених людей, в наші дні ця індустрія породила багато спеціальностей, в яких застосовані тисячі людей. І розвиток цієї галузі триває. А для поповнення віртуальних світів, створюючи контент, потрібні різноманітні об'єкти. Кожна 3D модель, створена вручну, потребує великих трудозатрат. Їх можна створювати, використовуючи процедурну генерацію, а можна відновлювати існуючі в реальному світі об'єкти.

В комп'ютерній графіці існують різноманітні представлення об'єктів. Саме просте та часто використовуване – полігональна мережа (меш). Використовуючи її, можна створювати будь-яку поверхні з заданою точністю. Чем складніша поверхні, та більше полігонів потрібно використовувати в мережі. Існує одно заспособлення об'єктів – функція відстані з позначкою (SDF). Вона за координатами точок повертає відстань до поверхні об'єкта. Свою допомогою можна точно зобразити як тривимірні, так і двовимірні об'єкти.

Модель можна побудувати процедурно: об'єкти задаються за допомогою параметрів посередством процедурного генератора. Процедурний генератор – це функція, яка приймає на вхід параметри та повертає певне зображення об'єкта. Параметри можуть задавати характеристики об'єкта, інтуїтивно зрозумілі користувачеві, наприклад, розміри. Зберігання таких моделей займає значно менше пам'яті.

В процесі роботи реалізовані процедурні диференціювані генератори. Диференціюваність генератора означає, що є можливість знати залежність, яку повертає генератор від його параметрів, та зможе зробити частину производною. Для різних зображенняв необхідно зробити производні по-різному. Для полігональної мережі – производну позиції кожного вузла мережі за параметрами. Для функції відстані з позначкою – производну відстані за параметрами. Наличчя у генератора производних дозволяє оптимізувати його результат. Таким чином, з'являється можливість розв'язання оберненої задачі – отримання параметрів для генератора за моделлю або її частиною.

Диференціювані представлення об'єктів та диференціювані процедурні генератори дають можливість отримувати моделі, використовуючи градієнтні методи оптимізації. Якщо ж генератори будуть ще й універсальними, це може дозволити реконструювати будь-які моделі, не приділяючи уваги виду або класу об'єкта. Написані в рамках цієї роботи генератори вже використовуються для відновлення об'єктів за однім зображенням [1] [2].

Ідея універсального генератора полягає в тому, що з його допомогою можна створювати будь-яку модель. Тобто, не потрібно створювати принципіально новий генератор, якщо з'явиться необхідність в об'єктах нового класу. Але реконструкція з використанням такого генератора буде потребувати більш складного алгоритму.

2 Постановка задачи

Основная задача состояла в разработке дифференцируемых процедурных генераторов и, затем, универсального дифференцируемого процедурного генератора.

Процедурный генератор должен по списку параметров длины n возвращать модель в одном из существующих представлений. В данном случае, возвращать либо полигональную сетку, либо функцию расстояния со знаком.

Поскольку генератор дифференцируемый, в дополнение к модели он должен возвращать производные составляющих модели по входным параметрам.

2.1 Цели работы

Были поставлены следующие цели работы:

1. Разработать дифференцируемый процедурный генератор моделей 1-ого типа;
2. Разработать дифференцируемый процедурный генератор моделей 2-ого типа;
3. Интегрировать генераторы в программу 3D реконструкции;
4. Реализовать составляющие универсального дифференцируемого процедурного генератора;
5. Произвести экспериментальную оценку результатов реконструкции с использованием дифференцируемых процедурных генераторов.

2.2 Формальная постановка задачи

В работе созданы генераторы, использующие разные представления объектов:

1. Полигональные сетки:

Вход:

- $Params \in \mathbb{R}^n$ - вектор длины n , набор числовых параметров для генерации;
- $Structure \in \mathbb{N}^m$ - вектор длины m , список, описывающий структуру дерева процедурного генератора;
- $n \in \mathbb{N}$ - число параметров;
- $m \in \mathbb{N}$ - число узлов в дереве.

Выход:

- $Mesh \in \mathbb{R}^{3 \cdot k}$ - массив длины $3 \cdot k$, полигональная сетка, модель полученная из параметров;
- $Jac \in \mathbb{R}^{(3 \cdot k) \times n}$ - матрица размера $3 \cdot k \times n$, содержащая производные элементов полигональной сетки по входным параметрам;
- $k \in \mathbb{N}$ - число точек в сетке.

2. Функции расстояния со знаком:

Вход:

- $Params \in \mathbb{R}^n$ - вектор длины n , набор числовых параметров для генерации;
- $Position \in \mathbb{R}^3$ - вектор длины 3, позиция, в которой будет посчитан результат функции;
- $Structure \in \mathbb{N}^m$ - вектор длины m , список, описывающий структуру дерева процедурного генератора;
- $n \in \mathbb{N}$ - число параметров;
- $m \in \mathbb{N}$ - число узлов в дереве.

Выход:

- $Dist \in \mathbb{R}$ - действительное число, знаковое расстояние до поверхности модели, полученное по входным параметрам в заданной точке;
- $Jac \in \mathbb{R}^{1 \times (n+3)}$ - матрица размера $1 \times (n + 3)$, содержащая производные дистанции по входным параметрам и входной позиции.

3 Обзор литературы

3.1 Представления объектов

В компьютерной графике существует большое количество представлений объектов, таких как:

- Полигональные сетки;
- Воксельные сетки;
- Функции расстояния;
- Облака точек;
- Процедурные представления;
- Неявные представления, и другие.

Представление объектов в виде полигональной сетки является наиболее распространённым. Оно позволяет легко переносить различные модели между многими приложениями для редактирования или использования, например Blender, AutoCAD, и многие другие. Достаточно просто можно добавить мелкие детали и изменить форму объекта. Также есть возможность применять эффективные алгоритмы их рендеринга. Но при этом существуют и недостатки. Так, оптимизация поверхности модели может быть сложной, из-за невыпуклости оптимизируемой функции, в особенности при наличии мелких деталей [1] [2]. Чтобы попробовать решить возникшую проблему, можно использовать воксельные сетки [3] [4]. Они помогут упростить поиск экстремума. Но и этот способ представления сцены имеет свои минусы. Восстановление объектов с мелкими деталями сильно усложняется. В качестве альтернативы предыдущим представлениям можно рассмотреть облака точек. Этот способ позволяет создавать высококачественные реконструкции сцен [5] [6].

На сегодняшний день нейронные сети являются одним из самых актуальных направлений в совершенно различных областях. И в данном обзоре, безусловно, нельзя обойтись без их упоминания. Тем более, что использование нейронных сетей на основе координат, или по другому – нейронные поля, даёт довольно интересные результаты [7] [8] [9]. Нейронные поля обрабатывают сложные сцены за считанные секунды, при этом выдавая достаточно качественные результаты. Однако, часто, их бывает неудобно использовать. Это происходит из-за того, что на практике эти результаты необходимо преобразовать из неявного представления в полигональную сетку, что даёт некачественную и неоптимальную модель.

3.2 Процедурная генерация

Процедурное представление объектов отсылает к большой области – процедурной генерации. Процедурная генерация контента – это способ создания объектов совершен-

но разных классов, от простых предметов, до целых виртуальных миров, при помощи алгоритмов и функций [10] [11]. Этой теме посвящено много разнообразных статей [12] [13]. Кроме того, стоит упомянуть об обратной процедурной генерации, которая может быть использована для восстановления параметров модели [14] [15] [16] [1] [2]. Поскольку сложные процедурные генераторы часто являются необратимыми функциями, работы по этой теме, несмотря на их большое количество и многообразие, плохо обобщаются, и обычно сосредоточены на конкретных классах процедурных моделей.

Процедурные генераторы могут быть созданы с использованием формальных грамматик [17] [18]. Также интересным, и даже более распространённым является использование так называемых систем Линденмайера (L -систем), которые являются специальным видом формальной грамматики [19] [15]. Такие генераторы описывают модель при помощи набора правил преобразований символов из «алфавита» и начального символа. С использованием таких методов был реализован обратный процедурный генератор, основанный на L -системах, о чём подробно изложено в следующей работе [20]. Он позволяет задать L -системой любое двухмерное векторное изображение, а также по входному векторному изображению получить L -систему и легко её модифицировать при необходимости.

Интересной представляется возможность создания генераторов на основе деревьев или графов, например, как в этой статье [21]. В ней представлен генератор в виде графа узлов, с элементами дифференцирования. Однако, эта реализация не предполагает использование генератора для оптимизации за пределами некоторой локальной области, что отличает её от генераторов, описанных в рамках данной квалификационной работы.

3.3 Реконструкция трёхмерных моделей

Дифференцируемые генераторы создавались с целью использования их в задаче реконструкции трёхмерных объектов по одному изображению. Эта задача, на сегодняшний день, остается очень сложной, в первую очередь потому, что она является существенно некорректной. Очевидно, что на одном изображении видна только часть объекта, а для восстановления невидимых частей необходимо использовать дополнительные знания о структуре реконструируемого объекта. По этой причине методы машинного обучения являются самой распространенной группой методов для решения этой задачи. Эти методы используют различные представления сцены, такие как полигональные сетки (меши) [22] [23] [24], воксельные сетки [25] [26], облака точек [27] [28] или неявные функции [29]. Важно отметить, что эти методы обучаются и оцениваются на одних и тех же классах объектов и их применение к другим классам требует повторного обучения и наличия достаточно большой выборки 3D-моделей соответствующего типа. Недавняя работа [30] показала, что такие подходы к реконструкции, в первую очередь, выполняют распознавание типа объекта, а не его реконструкцию и повышение качества результата их работы сильно затруднительно.

Также стоит обратить внимание на несколько работ, реализующих восстановление 3D-модели по одному изображению, не ограничивая категорию объекта [31] [32], что, в свою очередь, приводит к более низкому качеству моделей тех классов, которые не присутствовали в обучающей выборке.

Работы, связанные с процедурными генераторами и с обратной процедурной генерацией, в большинстве своём направлены на решение какой-то конкретной узкой задачи и плохо обобщаются. В то время, как эта работа посвящена созданию универсального дифференцируемого процедурного генератора, который планируется использовать для решения задачи 3D реконструкции без концентрации на конкретном классе объектов, рассматривая данную проблему значительно шире.

4 Предложенные методы

4.1 Генераторы посуды и зданий

4.1.1 Подсчёт производных

Дифференцируемый процедурный генератор отличается от обычного тем, что помимо созданного с помощью параметров объекта, он должен вернуть матрицу производных, в которой описывается зависимость всех выходных элементов от входных не дискретных параметров. Эта матрица может использоваться в алгоритмах реконструкции для расчёта производных функции потерь по параметрам [1] [2]:

$$\frac{\partial Loss}{\partial P} = \frac{\partial Loss}{\partial Pos} \cdot \frac{\partial Pos}{\partial P}$$

Стоит заметить, что в большом количестве нетривиальных процедурных генераторов существуют дискретные зависимости, например, количество окон у здания, или наличие ручки у чашки. Для подбора этих параметров не получится использовать производные, и придётся искать другие подходящие алгоритмы, такие как, например, меметический алгоритм (memetic algorithm) [1] [2].

4.1.2 Процедурный генератор посуды

Первый из разработанных дифференцируемых процедурных генераторов – процедурный генератор посуды. Он является примером простого генератора, близкого в своем поведении к дифференцируемой функции, и имеет всего один дискретный параметр. Все остальные параметры у него вещественные.

Алгоритм создания полигональной сетки выглядит следующим образом:

- Создаётся ломанная линия – сплайн по конечному числу параметров, имеющая координаты углов линии: $(0, 0), (x_1, 0), (x_2, 1) \dots (x_n, n - 1)$;

- Линия загибается таким образом, чтобы получилась замкнутая петля определённой ширины, которая задаётся в параметрах;
- Эта линия вращается вокруг оси ОY конечное количество раз, создавая дискретное тело вращения, и для каждой пары линий создаются треугольники в углах линий, которые добавляются в полигональную сетку;
- Если соответствующий дискретный параметр не равен 0, то создаётся правильный многоугольник, который несколько раз проворачивается на постоянный угол относительно центра, и отодвигается от центра на расстояние, заданное в параметрах для каждого поворота отдельно. Далее для каждой пары соседних многоугольников вновь создаются треугольники, соединяющие углы, и записанные в итоге в полигональную сетку.

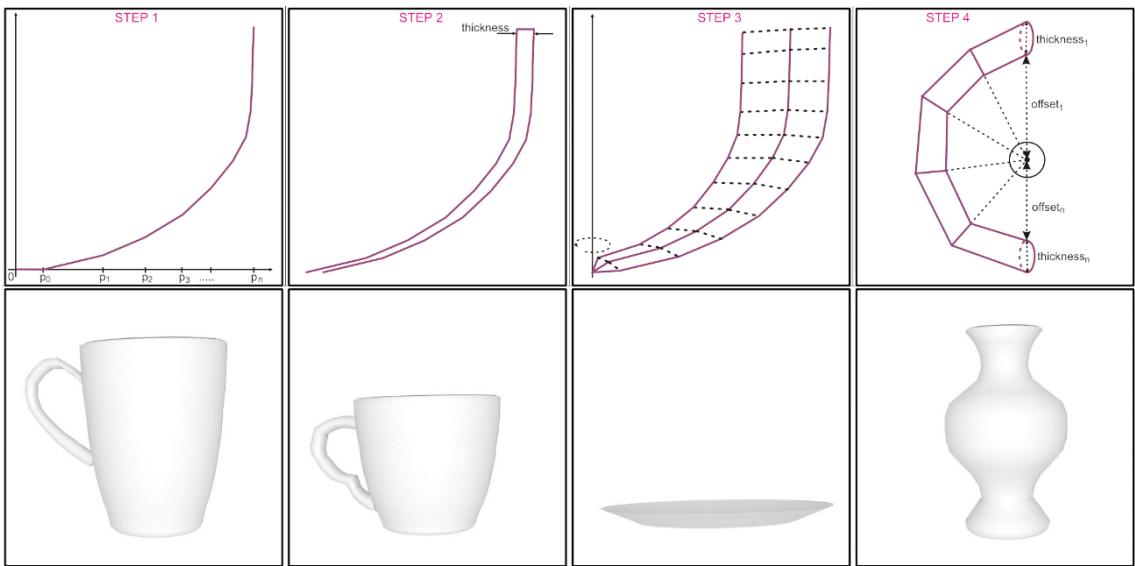


Рис. 1: Верхний ряд: этапы генерации. Слева направо: 1) Создание сплайна из вектора вертикальных смещений. 2) Преобразование для создания замкнутого сплайна с заданной толщиной. 3) Вращение. 4) (Необязательно) Создание ручки из сплайна окружности и вектора смещений. Количество точек в сплайнах может быть изменено для получения различных уровней детализации на основе одного и того же набора параметров. Нижний ряд: примеры сгенерированных объектов. Слева направо: 1) Кружка 2) Чайная чашка 3) Тарелка 4) Ваза.

Для поиска производных использовалась библиотека автоматического дифференцирования CppAD [33], которая позволяет получать требуемые производные из вычислительного графа, не вычисляя их явно внутри кода.

4.1.3 Процедурный генератор зданий

Следующий реализованный генератор – генератор панельных зданий. В нём существенно больше дискретных параметров, отвечающих за количество этажей и подъездов

и т.д. При этом он также обладает и вещественными параметрами, по которым возможно автоматическое дифференцирование. При создании полигональной сетки каждый элемент здания создаётся отдельно и добавляется в общий массив полигонов.

- Стены с одной из сторон;
- Окна для каждой из стен;
- Стены с оставшимися сторонами;
- Окна и двери для вновь созданных стен;
- Балконы для каждого уже созданного окна;
- Крыша, и прочие мелкие детали.

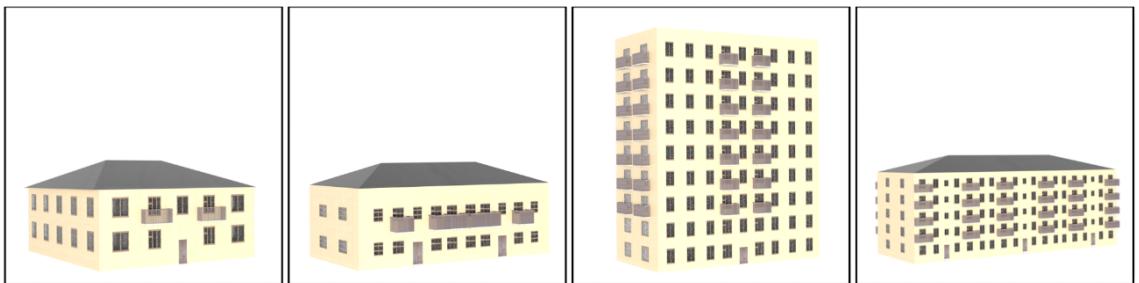


Рис. 2: Примеры созданных генератором зданий.

Для поиска производных по не дискретным параметрам в этом генераторе также использовалась библиотека CppAD [33].

4.2 Универсальные генераторы

4.2.1 Структура универсального генератора

Для реализации универсального генератора было написано дерево из узлов, в которых создавались примитивные модели, в дальнейшем модифицированные. Были реализованы узлы, создающие два типа моделей: полигональные сетки и функции расстояния со знаком. В процессе работы над универсальным генератором предпочтение было отдано реализации с использованием второго представления, так как это позволяло осуществлять более сложные взаимодействия между несколькими примитивами и упрощало расчёты, связанные с производными.

Узлы имеют функцию, которая выдаёт в качестве результата модель. У каждого узла разное количество параметров, от которых зависит результат функции расчёта модели. Также узлы имеют некоторое количество дочерних узлов (обычно 0 – 2, но может быть и более), от которых тоже зависит результат. Один из узлов дерева считается корневым: с него начинается расчёт модели, которую должен создать генератор. Каждый узел принимает на вход также указатели на массивы частных производных $\frac{\partial Dist}{\partial Pos}$ и

$\frac{\partial Dist}{\partial Params}$ и вычисляет их, если указатели не нулевые. Более подробно производные будут описаны в соответствующем разделе.

4.2.2 Виды узлов в дереве генератора

Дерево генератора состоит из узлов нескольких типов. Основные из них это:

- Узлы примитивов;
- Узлы преобразований;
- Узлы комбинаций.

Есть и другие, например, сложные узлы, но про них будет написано ниже. На рисунке 3 показаны виды узлов, и их вложенность.

Узлы примитивов – это те узлы, в которых высчитывается непосредственно некоторая примитивная модель от каких-то параметров. Они не зависят ни от каких других узлов, и, соответственно, не имеют дочерних узлов. Примеры узлов примитивов: узел шара, узел куба, узел цилиндра.

Узлы преобразований – это узлы, в которых происходит изменение одной модели, полученной из дочернего узла. Примеры узлов преобразований: узел масштабирования, узел сдвига, узел поворота.

Узлы комбинаций – те, в которых две других модели дают одну модель по каким-то правилам. Они зависят от двух дочерних узлов. Примеры подобных узлов: узел объединения, узел пересечения, узел вычитания.

Для каждого узла известно количество его дочерних узлов, и в связи с этим возможно однозначно представить дерево процедурного генератора как список узлов (например, в порядке обхода дерева: корень – левый узел – правый узел).

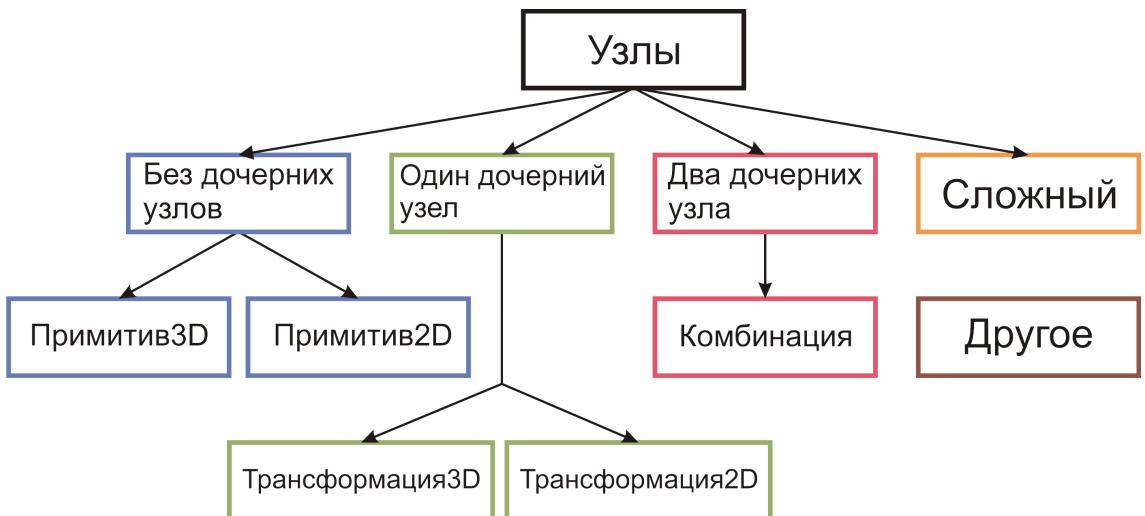


Рис. 3: Виды узлов и их вложенность.

4.2.3 Подсчёт производных в узлах

Для подсчёта производных в дереве генератора необходимо посчитать их в каждом из узлов, в зависимости от входных параметров, и выходных данных дочерних узлов, то есть от представления объекта и матрицы производных, выданных каждым из дочерних узлов.

Далее представлена формула подсчёта производных для узла, с n дочерними узлами, каждый из которых зависит от k_i параметров. Все параметры считаются независимыми друг от друга.

$$R = M \cdot U | N$$

В этом уравнении, операция \cdot является матричным умножением, а операция $|$ конкатацией двух матриц;

$$R = \begin{pmatrix} \frac{\partial new_1}{\partial P_{1,1}} & \cdots & \frac{\partial new_1}{\partial P_{1,k_1}} & \cdots & \frac{\partial new_1}{\partial P_{n+1,k_{n+1}}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial new_m}{\partial P_{1,1}} & \cdots & \frac{\partial new_m}{\partial P_{1,k_1}} & \cdots & \frac{\partial new_m}{\partial P_{n+1,k_{n+1}}} \end{pmatrix}$$

Матрица R содержит результат, то есть производные выходной модели по всем параметрам;

$$M = \begin{pmatrix} \frac{\partial new_1}{\partial old_{1,1}} & \cdots & \frac{\partial new_1}{\partial old_{1,m_1}} & \cdots & \frac{\partial new_1}{\partial old_{n,m_n}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial new_m}{\partial old_{1,1}} & \cdots & \frac{\partial new_m}{\partial old_{1,m_1}} & \cdots & \frac{\partial new_m}{\partial old_{n,m_n}} \end{pmatrix}$$

Матрица M содержит зависимости новой модели от дочерних, то есть производные элементов выходной модели по элементам всех входных моделей;

$$U = \begin{pmatrix} \frac{\partial old_{1,1}}{\partial P_{1,1}} & \cdots & \frac{\partial old_{1,1}}{\partial P_{1,k_1}} & \cdots & \frac{\partial old_{1,1}}{\partial P_{n+1,k_{n+1}}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial old_{1,m_1}}{\partial P_{1,1}} & \cdots & \frac{\partial old_{1,m_1}}{\partial P_{1,k_1}} & \cdots & \frac{\partial old_{1,m_1}}{\partial P_{n+1,k_{n+1}}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial old_{n,m_n}}{\partial P_{1,1}} & \cdots & \frac{\partial old_{n,m_n}}{\partial P_{1,k_1}} & \cdots & \frac{\partial old_{n,m_n}}{\partial P_{n+1,k_{n+1}}} \end{pmatrix}$$

Матрица U содержит зависимости всех дочерних моделей от всех старых параметров, то есть производные элементов всех входных моделей по параметрам, от которых они зависят;

$$N = \begin{pmatrix} \frac{\partial new_1}{\partial P_{n+1,1}} & \cdots & \frac{\partial new_1}{\partial P_{n+1,k_{n+1}}} \\ \vdots & \ddots & \vdots \\ \frac{\partial new_m}{\partial P_{n+1,1}} & \cdots & \frac{\partial new_m}{\partial P_{n+1,k_{n+1}}} \end{pmatrix}$$

Матрица N содержит зависимости полученной модели от новых параметров, то есть производные элементов выходной модели по параметрам, от которых не зависят дочер-

ние модели.

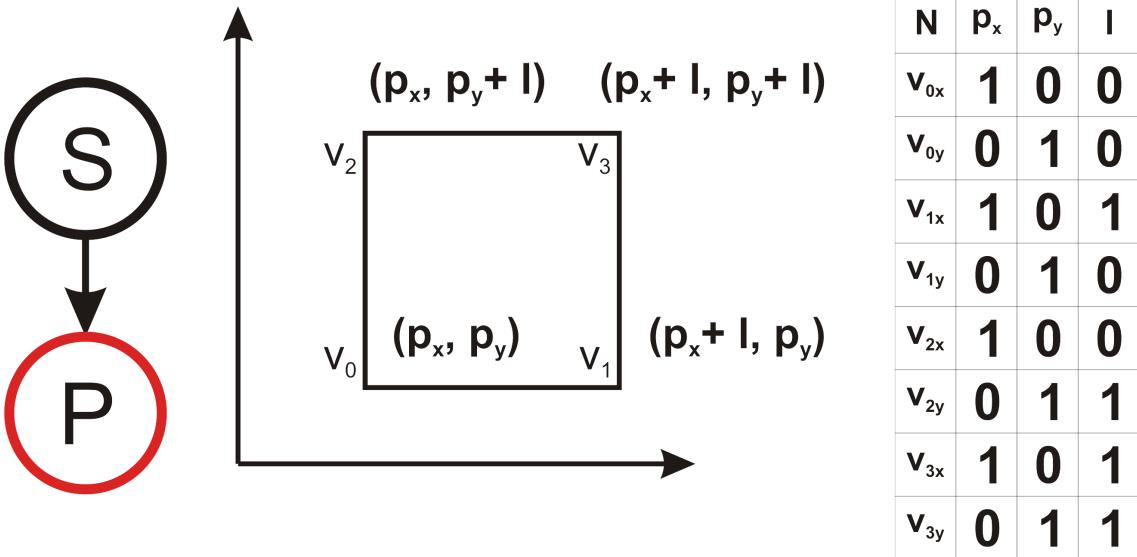
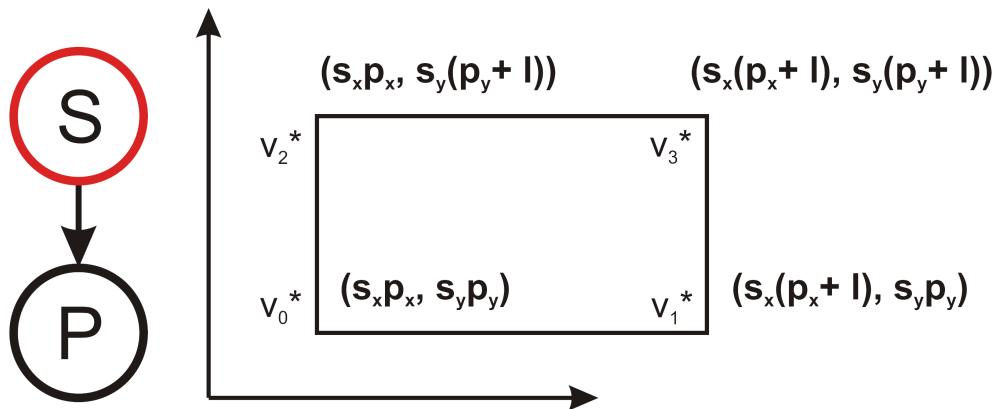


Рис. 4: Вычисления матрицы производных в узле примитива.

В качестве примера выше изложенного, можно предложить рассмотреть простое дерево, состоящее из двух узлов. На изображении 4 находится первый узел, который создаёт квадрат заданного размера l с нижним левым углом в заданной координате p . На изображении 5 представлены вычисления в узле масштабирования в s_x раз по оси x , и в s_y по оси y . В формуле узла примитива матрицы M и U будут пустыми, по причине отсутствия дочерних узлов. Матрица N будет иметь размер содержать зависимости восьми координат углов квадрата от трёх входных параметров. Итоговая матрица R в этом узле будет совпадать с матрицей N . Затем необходимо посчитать матрицу зависимостей уже в узле преобразований. Матрица M будет содержать зависимость новых координат углов квадрата от старых координат. Поскольку каждая координата угла меняется по формуле: $v'_i = v_i * s$, то и матрица M будет выглядеть как диагональная матрица, содержащая на диагонали s_x и s_y . Матрица U будет совпадать с итоговой матрицей в узле примитива из-за того, что дочерний узел один. N , в свою очередь, содержит зависимость изменённых углов по новым переменным. В результате каждая строка матрицы U увеличится в s_x или s_y раз, и к ней будет приписана новая матрица N , откуда и получается матрица зависимостей R в узле преобразований.



M	v_{0x}	v_{0y}	v_{1x}	v_{1y}	v_{2x}	v_{2y}	v_{3x}	v_{3y}
v_{0x}*	s_x							
v_{0y}*		s_y						
v_{1x}*			s_x					
v_{1y}*				s_y				
v_{2x}*					s_x			
v_{2y}*						s_y		
v_{3x}*							s_x	
v_{3y}*								s_y

U	p_x	p_y	I
v_{0x}	1	0	0
v_{0y}	0	1	0
v_{1x}	1	0	1
v_{1y}	0	1	0
v_{2x}	1	0	0
v_{2y}	0	1	1
v_{3x}	1	0	1
v_{3y}	0	1	1

N	s_x	s_y
v_{0x}*	v_{0x}	0
v_{0y}*	0	v_{0y}
v_{1x}*	v_{1x}	0
v_{1y}*	0	v_{1y}
v_{2x}*	v_{2x}	0
v_{2y}*	0	v_{2y}
v_{3x}*	v_{3x}	0
v_{3y}*	0	v_{3y}

Рис. 5: Вычисления матрицы производных в узле преобразований.

Перемножение матриц – это затратная операция, что очень сильно может сказаться на скорости расчёта матрицы производных. Поэтому, если представление объекта содержит много элементов, было бы целесообразно уменьшить количество необходимых расчётов при получении новой матрицы производных.

В случае с деревом, создающим объект как полигональную сетку, количество элементов в ней может быть крайне большим, что разумеется скажется на времени работы. Поэтому было решено отбросить из рассмотрения операции, не имеющие большого смысла, но при этом создающие матрицы, сложной структуры. В итоге все оставшиеся виды узлов отличались тем, что в матрице производных нового объекта по старому объекту ненулевые производные могли находиться только в подматрицах 3×3 , описывающих зависимости точки после изменения узлом от точки до изменения узлом. Это заметно упрощало необходимые расчёты и позволяло считать матрицу производных даже при наличии большого количества полигонов в создающемся объекте.

Если же в генераторе использовать представление объекта – функцию расстояния со знаком, то матрица зависимостей нового объекта от старых вырождается в вектор, что уменьшает количество вычислений ещё больше. Если же рассматривать уже реализованные узлы, то в «узлах комбинаций» этот вектор состоит из констант -1, 0 и 1,

а в «узлах преобразований» вектор становится длины 1.

Для расчёта производных использовалась библиотека автоматического дифференцирования Enzyme AD [34] [35] [36].

4.2.4 Сложные узлы

В универсальном генераторе был создан класс сложных узлов. Он позволяет реализовывать новые узлы, используя старые, уже существующие, описывая поддерево из них, и отдельно – зависимость всех внутренних параметров поддерева от заявленных внешних параметров. Сложные узлы могут быть полезны для описания структур из большого количества простых объектов, у которых должны быть общие параметры. Для описания нового сложного узла необходимо записать в сложный узел структуру его поддерева и график зависимостей внутренних параметров поддерева, от параметров сложного узла. Примеры структур поддеревьев представлены на рисунках 6 и 10. Примеры графов зависимостей проиллюстрированы на рисунках 6 и 12.

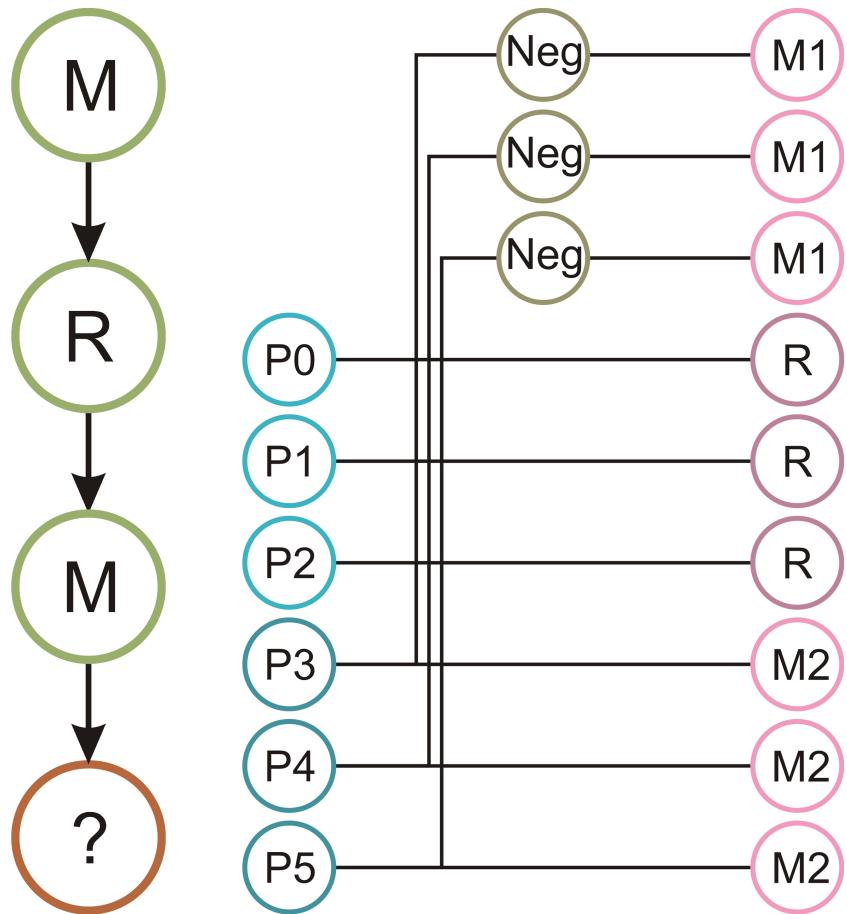


Рис. 6: Слева: Поддерево узлов, реализующих сложный узел «поворот относительно точки». “R” - узел простого поворота, “M” - узел перемещения, “?” - выход поддерева, к которому будет присоединён дочерний узел. Справа: Зависимость внутренних параметров узла «поворот относительно точки» от внешних. Внешние параметры $P_0 - P_5$, графовые узлы отрицания Neg . Все остальные графовые узлы - внутренние параметры данного узла.

В этих узлах, разумеется, тоже необходимо считать производные. В поддереве производные считаются также, как и обычно, но после подсчёта производных в корне поддерева, необходимо получить зависимости от внешних параметров, по зависимостям от внутренних параметров. Для этих целей граф зависимостей внутренних параметров от внешних тоже считает свою матрицу производных. Из матрицы поддерева выделяется подматрица зависимостей от внутренних параметров, и умножается на матрицу, полученную из вычислительного графа. Результат вписывается в соответствующее место незатронутой матрицы производных, и получается нужная матрица зависимостей сложного узла.

4.2.5 Реализованный функционал

В универсальном дифференцируемом генераторе функций расстояния со знаком уже реализовано примерно двадцать пять узлов, описывающих различные распространённые примитивы и действия над ними. Одними из первых были реализованы следующие узлы примитивов:

- Шар;
- Параллелепипед;
- Цилиндр;
- Скруглённый параллелепипед;
- Призма;
- Конус;

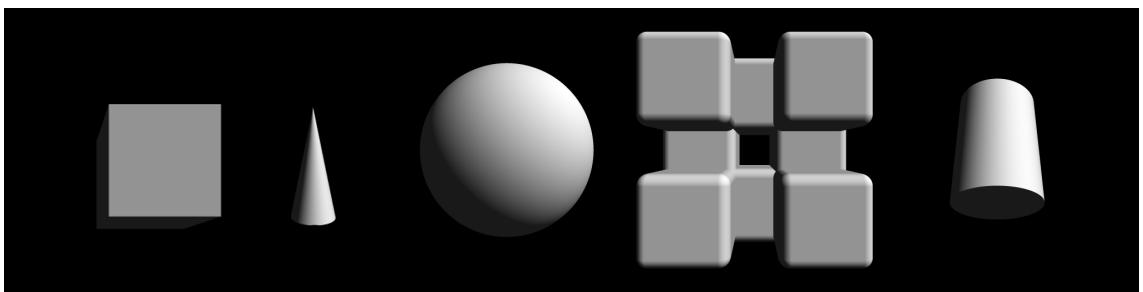


Рис. 7: Слева направо: узел - параллелепипед, узел - конус, узел - шар, комбинация нескольких узлов, каждый из которых является узлом - скруглённым параллелепипедом, повёрнутый узел - цилиндр.

Узлы преобразований:

- Перемещение;
- Масштабирование;

- Скругление;

Узлы взаимодействий:

- Объединение;
- Пересечение;
- Вычитание.

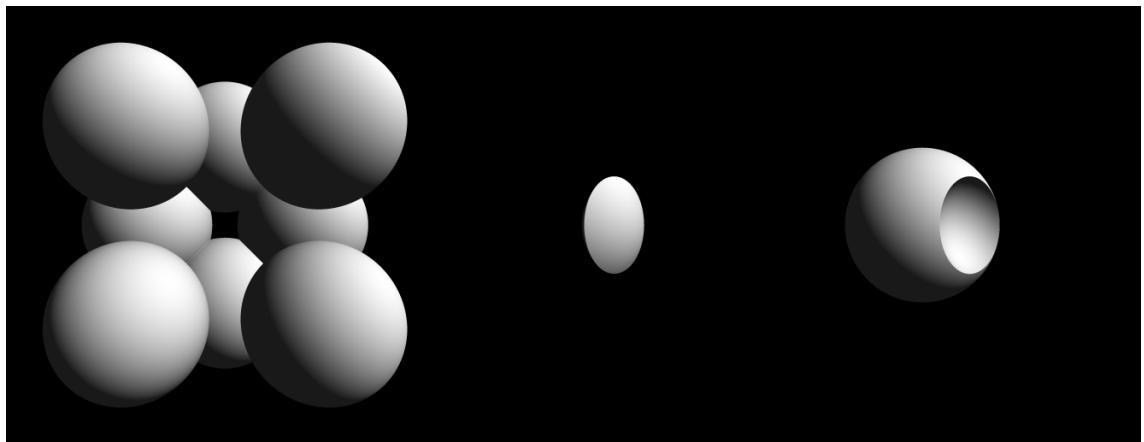


Рис. 8: Слева направо: демонстрация работы узла объединение, узла пересечение, узла вычитание.

Первым нетривиальным реализованным узлом преобразований оказался узел поворота. В нём в дочерний узел передаются изменённые параметры позиции, которые вычисляются умножением на матрицу поворота. Матрица, в свою очередь, вычисляется через три параметра: угол поворота, и два угла, задающие наклон оси вращения. Для получения производных в узле сначала вычисляются зависимости матрицы поворота от входных параметров, и через неё вычисляются необходимые производные объекта по входным параметрам, и объекта по неизменённой позиции.

Следующими были реализованы несколько сложных узлов: стул и поворот относительно точки. Для них был реализован класс сложных узлов, и записаны соответствующие поддеревья и графы зависимости параметров. Их графические представления изображены на рисунках 6, 10, 12. Результат совместной работы этих узлов показан на рисунке 9.

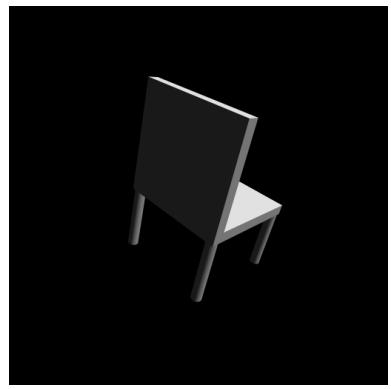


Рис. 9: Сложный узел поворота относительно точки, применённый к сложному узлу «стул».

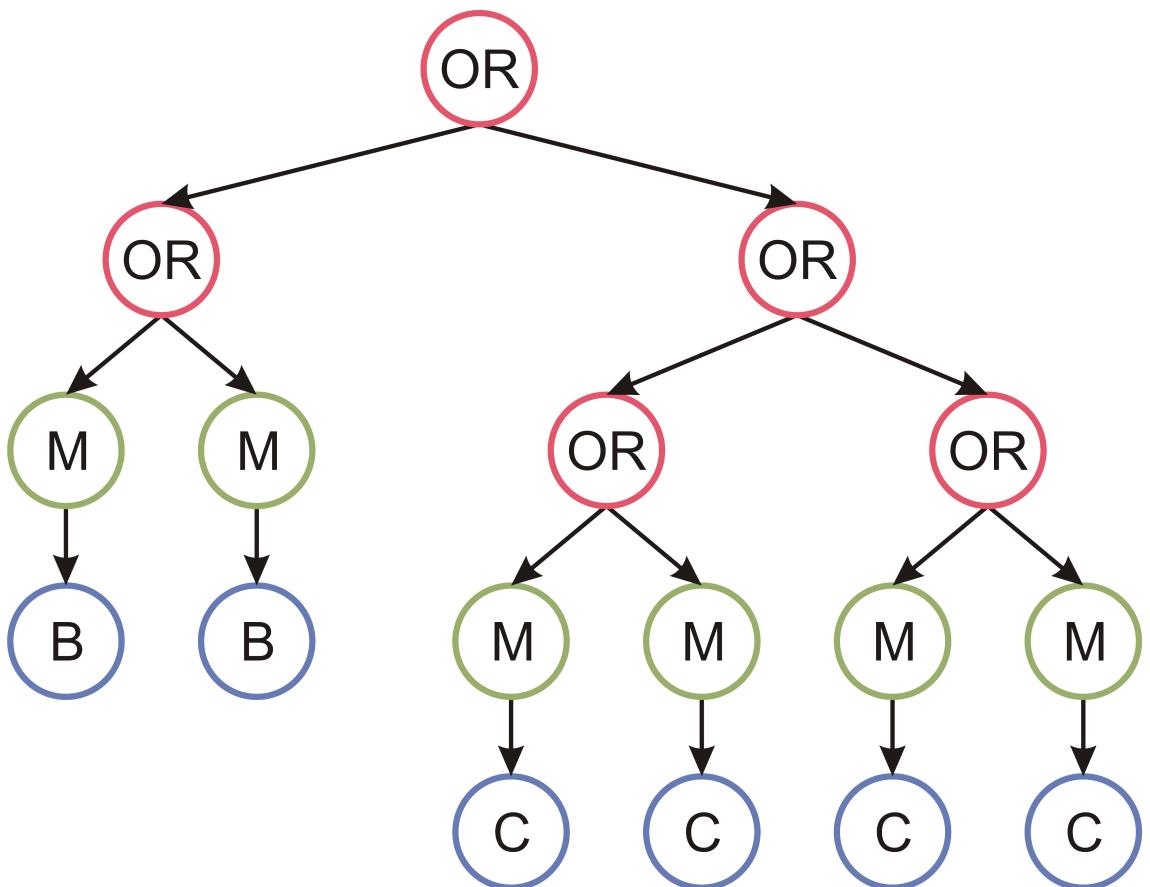


Рис. 10: Поддерево узлов, реализующих сложный узел «стул». “OR” - узел объединения, “M” - узел перемещения, “B” - примитивный узел - параллелепипед, “C” - примитивный узел - цилиндр.

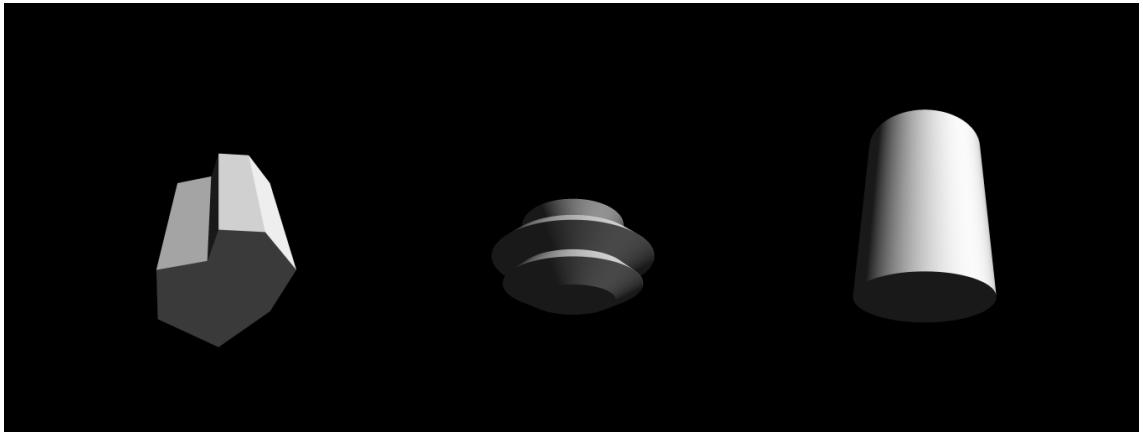


Рис. 11: Демонстрация работы, слева направо: 1) узла «экструзии», применённого к узлу «роза ветров», 2) узла «тело вращения», 3) узла «экструзии», применённого к узлу - круг (даёт результат, аналогичный использованию узла цилиндр).

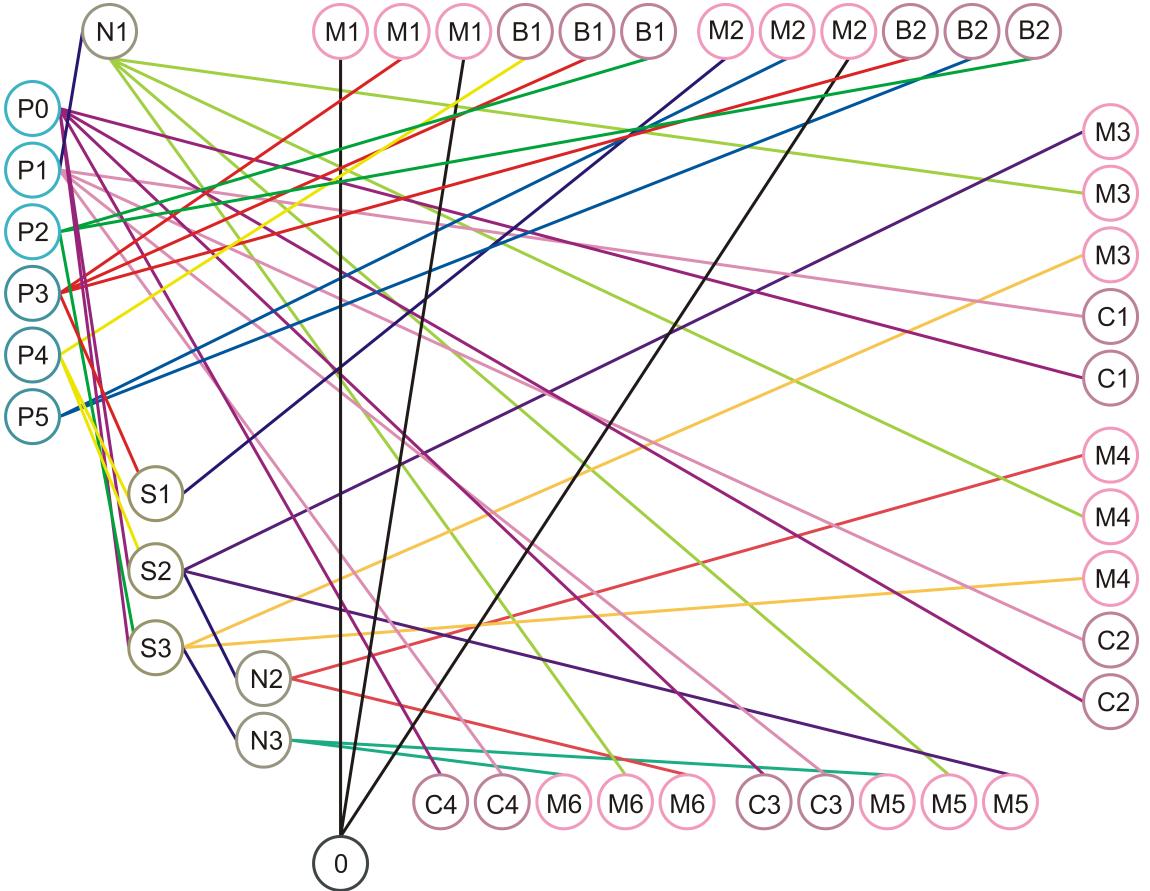


Рис. 12: Зависимость внутренних параметров узла «стул» от внешних. Внешние параметры $P_0 - P_5$, графовые узлы вычитания $S_1 - S_3$, графовые узлы отрицания $N_1 - N_3$, графовый узел константа 0. Все остальные графовые узлы - внутренние параметры данного узла.

Следующим нетривиальным узлом преобразований был узел «экструзии». Это узел, который двумерный объект превращает в трёхмерный, у которого задана высота, и основание совпадает с дочерним двумерным объектом. В начале происходит проверка,

откуда измеряется расстояние до объекта (сбоку от него, сверху, по диагонали, или внутри), и в зависимости от этого вычисляется результат и необходимые производные. Демонстрация работы этого узла представлена на рисунке 11.

Специально для корректного взаимодействия с узлом «экструзии» были созданы несколько узлов, описывающих двухмерные объекты:

- Квадрат;
- Круг;

И взаимодействия над ними в плоскости:

- Перемещение;
- Поворот;
- Поворот относительно точки.

Узлы масштабирования и скругления не были переписаны для двухмерных объектов, поскольку вычисления в них и количество входных параметров полностью повторяют соответствующие им узлы преобразования в трёхмерном пространстве.

Среди нетривиальных узлов также были созданы следующие узлы примитивов:

- Роза ветров (полигон, задающийся точками на границах секторов);
- Сплайн (ломанная линия);
- Тело вращения (ломанная линия, повёрнутая вокруг оси).

Они реализованы похожим образом, и отличаются друг от друга небольшими внутренними особенностями. Внутри каждого из них производится поиск ближайшего отрезка, относительно заданной позиции. После этого вычисляется расстояние до поверхности объекта и необходимые производные по параметрам, задающим этот отрезок, и другим, значимым для объекта. Результаты работы этих узлов представлены на изображении 11.

Узлов, которые возможно реализовать в этом универсальном генераторе, крайне много, и он легко дополняется новыми элементами. Простые узлы создаются по аналогии с уже созданными. Необходимо реализовать функцию расчёта расстояния до объекта, заданного узлом, и пересчитать матрицу производных в соответствии с формулой расстояния. При необходимости реализовать нетривиальный объект создаётся узел на основе класса сложного узла, в который записывается уникальное поддерево из уже существующих простых узлов, и необходимые взаимодействия между входными параметрами, и параметрами поддерева.

5 Процедурная генерация в задаче реконструкции

5.1 Генераторы посуды и зданий

Дифференцируемые процедурные генераторы были реализованы с целью решения задач реконструкции [1] [2]. Реконструкция происходит по следующему алгоритму:

- Трёхмерная модель создаётся дифференцируемым процедурным генератором;
- Модель вместе с параметрами сцены подаётся на вход дифференцируемому рендеру;
- Для выходного изображения получается силуэт объекта;
- Из эталонной и созданной маски вычисляется функция потерь;
- Проводится обратное распространение ошибки и изменяются параметры по меметическому алгоритму для следующей итерации.

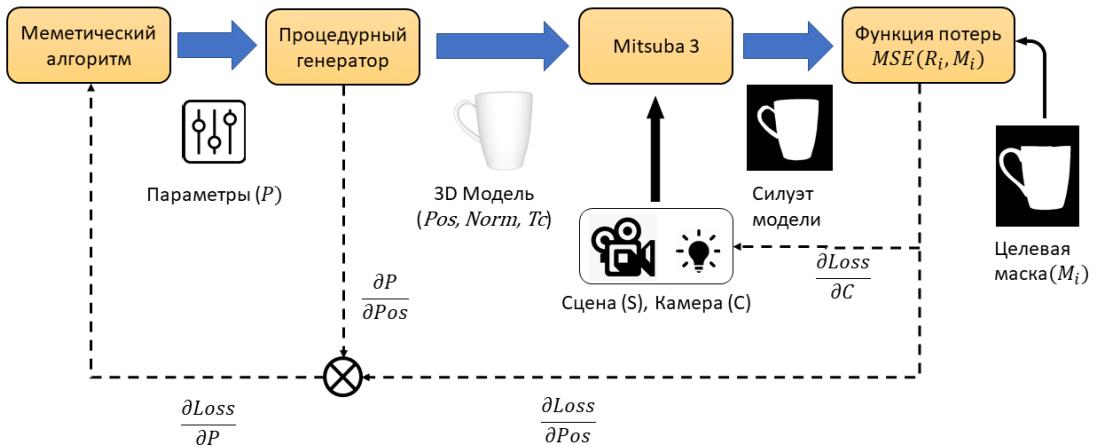


Рис. 13: Схема метода реконструкции модели.

Данный алгоритм использует дифференцируемые генераторы посуды и зданий. За счёт этого объект, полученный описанным выше методом, имеет более гладкую структуру и лучше восстанавливает форму входной модели. Процедурный генератор применяет строгие правила к структуре объекта и гарантирует, что полученная модель не будет содержать дыр, неровностей и иных геометрических артефактов, если они не предусмотрены структурой модели.



Рис. 14: Результаты реконструкции по одной фотографии с использованием процедурного генератора посуды. Верхний ряд: входные изображения, нижний ряд: реконструированные модели.



Рис. 15: Результаты реконструкции по одной фотографии с использованием процедурного генератора зданий. Слева направо: входное изображение, маска окон, результат реконструкции с использованием текстур по умолчанию.

На рисунках 14 и 15 показаны результаты реконструкции по одному входному изображению без информации о позиции и характеристиках камеры, а также освещении. На рисунке 16 и таблице 1 продемонстрировано сравнение восстановления модели с использованием дифференцируемого процедурного генератора, и других методов реконструкции.



Рис. 16: Сравнение результатов, полученных с использованием данного метода, с дифференцируемой реконструкцией SDF [37], InstantNGP [9] и Pixel2Mesh [22].

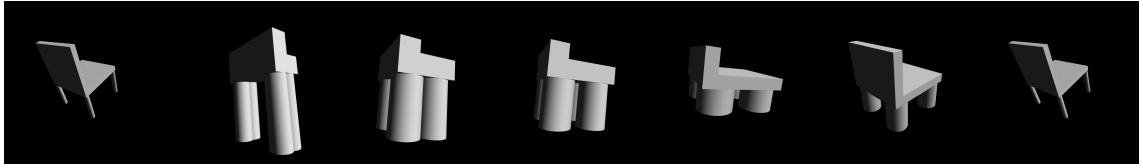


Рис. 17: Слева направо: целевая модель, исходная модель, модель после 100 итераций градиентного спуска, 200 итераций, 500 итераций, 750 итераций, оптимизированная модель (1000 итераций)

Алгоритм	cup_1	cup_2	building
DiffSDF (2 изображения)	0.787	0.422	0.728
DiffSDF (6 изображений)	0.960	0.502	0.967
DiffSDF (12 изображений)	0.984	0.669	0.979
InstantNGP (16 изображений)	0.858	0.878	0.940
InstantNGP (64 изображения)	0.905	0.930	0.971
Pixel2Mesh (1 изображение)	0.788	0.513	0.626
TGS (1 изображение)	0.939	0.925	0.965
Предложенный метод (1 изображение)	0.948	0.972	0.886
Предложенный метод (2 изображения)	0.951	0.978	0.890
Предложенный метод (4 изображения)	0.962	0.979	0.869

Таблица 1: Среднее значение IoU (больше - лучше) для различных моделей.

5.2 Универсальный генератор

Универсальный дифференцируемый процедурный генератор также может использоваться в задачах реконструкции трёхмерных объектов. При использовании фиксированного заранее заданного дерева генератора, объекты соответствующего класса могут восстанавливаться по схожему с выше изложенным алгоритму. На рисунке 17 проиллюстрирован процесс оптимизации модели градиентным спуском с фиксированным деревом.

В этом случае генератор не используется как универсальный. Оптимальным бы являлось использовать получать модели при помощи генератора, не зависимо от их класса, и без выбора одного конкретного дерева внутри него. В этом и состоит главная сложность создания универсального алгоритма реконструкции, что восстанавливаться должен не только объект, но и структура генератора. Но потенциал у использования универсального дифференцируемого генератора в задаче реконструкции есть. При его успешной реализации, художники и 3D моделисты смогут легко получать начальное приближение необходимого объекта и просто модифицировать структуру генератора или полученные параметры модели.

Произвести сравнение реализованных генераторов с другими обратными процедурными генераторами по какой-либо метрике не представляется целесообразным. Однако, можно попробовать сравнить важные в контексте данной работы характеристики некоторых обратных генераторов. В задаче трёхмерной реконструкции с использованием метода, представленного ранее, генератор должен быть дифференцируемым. Для реконструкции разных моделей, генератор должен восстанавливать как можно большее количество классов объектов, а в идеале быть универсальным. Данное сравнение продемонстрировано в таблице 2.

Алгоритм	Классы	Дискр. пар.	Дифф.
Proc. Architectural Models [17]	Здания	+	-
Inv. Proc. Mod. of L-systems [20]	2D изображения	+	-
Inv. Proc. Mod. of Branching Structures [15]	2D изображения	+	-
Inv. Proc. Mod. of Trees [14]	Деревья	+	-
Proc. Mod. of Buildings [38]	Здания	+	-
Предложенный метод тип 1	Посуда	+	+
Предложенный метод тип 2	Здания	+	+
Предложенный метод универсальный	Любые	+	+

Таблица 2: Важные характеристики разных процедурных генераторов. (Генерируемые классы объектов, Наличие дискретных параметров, Дифференцируемость)

6 Заключение

Данная работа посвящена созданию дифференцируемых процедурных генераторов, и реализации составляющих универсального дифференцируемого процедурного генератора 3D моделей. Дифференцируемость генераторов была использована в задаче трёхмерной реконструкции, а полученные результаты были проанализированы.

В рамках данной работы:

- Были реализованы дифференцируемые процедурные генераторы двух классов объектов, а конкретно, посуды и зданий;
- Генераторы были успешно интегрированы в программу 3D реконструкции;
- Результаты реконструкции, полученные с использованием дифференцируемых процедурных генераторов, были получены и произведена экспериментальная оценка;
- Также были разработаны составляющие универсальных дифференцируемых процедурных генераторов, с возможностью их дальнейшей модификации, использующих представления объектов как полигональные сетки, и как функции расстояния со знаком.

Результаты, полученные после интеграции генераторов в программу для реконструкции, были описаны в статье [1] [2], представленной на международной конференции ГрафиКон 2023 года и опубликованной в Q1 журнале “Symmetry”.

Список литературы

- [1] Albert Garifullin, Nikolay Maiorov, and Vladimir Frolov. Differentiable Procedural Models for Single-view 3D Mesh Reconstruction. In Peter Vangorp and David Hunter, editors, *Computer Graphics and Visual Computing (CGVC)*. The Eurographics Association, 2023.
- [2] Albert Garifullin, Nikolay Maiorov, Vladimir Frolov, and Alexey Voloboy. Single-view 3d reconstruction via differentiable rendering and inverse procedural modeling. *Symmetry*, 16(2):184, 2024.
- [3] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.
- [4] Delio Vicini, Wenzel Jakob, and Anton Kaplanyan. A non-exponential transmittance model for volumetric scene representations. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021.
- [5] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.
- [6] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.
- [7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [8] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.
- [9] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.
- [10] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22, 2013.
- [11] Jonas Freiknecht and Wolfgang Effelsberg. A survey on the procedural generation of virtual worlds. *Multimodal Technologies and Interaction*, 1(4):27, 2017.

- [12] Breno MF Viana and Selan R dos Santos. A survey of procedural dungeon generation. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 29–38. IEEE, 2019.
- [13] Daniel Michelon De Carli, Fernando Bevilacqua, Cesar Tadeu Pozzer, and Marcos Cordeiro d’Ornellas. A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian symposium on games and digital entertainment*, pages 26–35. IEEE, 2011.
- [14] Ondrej Stava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. Inverse procedural modelling of trees. In *Computer Graphics Forum*, volume 33, pages 118–131. Wiley Online Library, 2014.
- [15] Jianwei Guo, Haiyong Jiang, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, Dani Lischinski, and Hui Huang. Inverse procedural modeling of branching structures by inferring l-systems. *ACM Transactions on Graphics (TOG)*, 39(5):1–13, 2020.
- [16] Albert Garifullin, Alexandr Shcherbakov, and Vladimir Frolov. Fitting parameters for procedural plant generation. 2022.
- [17] Bedrich Benes Ilke Demir, Daniel G. Aliaga. Proceduralization for editing 3d architectural models. In *International Conference on 3D Vision (Proceedings 3DV 2016)*, page 9 pages. 3DV’16, 2016.
- [18] Francesco Sportelli, Giuseppe Toto, GENNARO VESSIO, et al. A probabilistic grammar for procedural content generation. In *Proceedings of the 6th Workshop on Non-Classical Models of Automata and Applications (NCMA 2014)-Short Papers*, pages 31–36. Oesterreichische Computer Gesellschaft, 2014.
- [19] Jean-Eudes Marvie, Julien Perret, and Kadi Bouatouch. The fl-system: a functional l-system for procedural geometric modeling. *The Visual Computer*, 21(5):329–339, 2005.
- [20] Ondrej Št’ava, Bedrich Beneš, Radomir Měch, Daniel G Aliaga, and Peter Krištof. Inverse procedural modeling by automatic generation of l-systems. In *Computer graphics forum*, volume 29, pages 665–674. Wiley Online Library, 2010.
- [21] Mathieu Gaillard, Vojtěch Krs, Giorgio Gori, Radomír Měch, and Bedrich Benes. Automatic differentiable procedural modeling. In *Computer Graphics Forum*, volume 41, pages 289–307. Wiley Online Library, 2022.
- [22] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.
- [23] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor

- scenes from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 55–64, 2020.
- [24] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. Shelf-supervised mesh prediction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8843–8852, 2021.
 - [25] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VIII 14*, pages 628–644. Springer, 2016.
 - [26] Stefan Popov, Pablo Bauszat, and Vittorio Ferrari. Corenet: Coherent 3d scene reconstruction from a single rgb image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 366–383. Springer, 2020.
 - [27] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.
 - [28] Chao Chen, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Unsupervised learning of fine structure generation for 3d point clouds by 2d projections matching. In *Proceedings of the ieee/cvf international conference on computer vision*, pages 12466–12477, 2021.
 - [29] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in neural information processing systems*, 32, 2019.
 - [30] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3405–3414, 2019.
 - [31] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. *Advances in neural information processing systems*, 31, 2018.
 - [32] Xianghui Yang, Guosheng Lin, and Luping Zhou. Zeromesh: Zero-shot single-view 3d mesh reconstruction. *arXiv preprint arXiv:2208.02676*, 2022.
 - [33] Bradley M Bell. Cppad: a package for c++ algorithmic differentiation. *Computational Infrastructure for Operations Research*, 57(10), 2012.

- [34] William Moses and Valentin Churavy. Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12472–12485. Curran Associates, Inc., 2020.
- [35] William S. Moses, Valentin Churavy, Ludger Paehler, Jan Hückelheim, Sri Hari Krishna Narayanan, Michel Schanen, and Johannes Doerfert. Reverse-mode automatic differentiation and optimization of gpu kernels via enzyme. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’21, New York, NY, USA, 2021. Association for Computing Machinery.
- [36] William S. Moses, Sri Hari Krishna Narayanan, Ludger Paehler, Valentin Churavy, Michel Schanen, Jan Hückelheim, Johannes Doerfert, and Paul Hovland. Scalable automatic differentiation of multiple parallel paradigms through compiler augmentation. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC ’22. IEEE Press, 2022.
- [37] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.
- [38] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623. 2006.