Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет Цифровых трансформаций

Образовательная программа Технологии разработки компьютерных игр

Направление подготовки (специальность) Прикладная информатика

О Т Ч Е Т

о научно-исследовательской работе

Тема задания: Разработка модели генерации планировочных решений для жилых помещений исходя из правил эргономики и эффективности использования пространства

Обучающийся Лепин Даниил Александрович, J4223

Руководитель практики от университета: Карсаков Андрей Сергеевич, доцент факультета цифровых трансформаций

Практика пройдена с оценкой _____

Подписи членов комиссии:

_____
(подпись)

_____
(подпись)

_____
(подпись)

Дата _____

Санкт-Петербург
2023

Реферат

Отчет о научно-исследовательской работе объемом 27 страниц содержит 12 рисунков, список использованных источников из 34 наименований.

Объектом исследования являются методы генерации планировочных решений в жилых помещениях.

Целью этого исследования является достижение более реалистичных результатов при генерации жилых помещений.

Задачей данного исследования является изучение существующих подходов генерации жилых помещений и описание критериев для оценки реалистичности генерации исходя из правил эргономики и эффективности использования пространства.

Отчет состоит из трех частей. В первой части приведен анализ предметной области, рассмотрены существующие подходы генерации планировочных решений, выявлены проблемы существующих подходов. Во второй части сформулированы выводы о проделанной работе. В третей части приведен план дальнейшей работы над проектом.

Метод исследования: аналитический.

Ключевые слова: Процедурная генерация, планировки, алгоритмы генерации, 3д модели, эргономика.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматическая генерация контента сейчас встречается во многих сферах развлечений, дизайна или архитектуры. Алгоритмы могут помочь сэкономить время проектировщику зданий или дизайнеру видеоигровых локаций при создании однотипного контента. Например, использование автоматически сгенерированного контента при производстве видеоигр давно стало нормой как среди больших компаний, так и малых коллективов. Подобные подходы можно встретить при создании ландшафтов для игр с открытым миром, создании множества разнообразных моделей людей с уникальными чертами или целых звездных систем. Также генерация контента может разнообразить игровой процесс пользователя предлагая ему уникальные ситуации, меняющиеся с каждым игровым циклом.

Темой этого исследования является генерация планировочных решений для жилых помещений. Такой тип генерации является перспективным направлением в сфере недвижимости. Несколько компаний разного размера в настоящее время сосредоточены на создании инструментов автоматической планировки, чтобы помочь застройщикам и инвесторам. Они могут помочь максимизировать потенциальную площадь застройки и выполнить автоматический технико-экономический анализ участка, который может информировать об инвестиционных возможностях для приобретения земли и максимизировать выгоду от арендуемой площади. Независимо от того, ориентированы ли они на индустрию недвижимости или задуманы как собственные программные инструменты в архитектурных и инженерных фирмах, большинство современных подходов решают задачи автоматизации планировки в масштабе отдельного здания.

Также генерация помещений может встречается в видеоиграх и этот тип контента никак не привязан к определенным игровым жанрам, он может применятся в шутерах, приключенческих играх, головоломках, ммо, стратегиях и др. Поэтому подобная модель генерации будет востребована во многих проектах.

Самыми важным аспектами при оценке эффективности алгоритма генерации являются следующие критерии: скорость генерации, гибкость настройки и реалистичность конечного результата. И если с критерием скорости все предельно просто – чем быстрее генерация, тем лучше, то критерий реалистичности результата может вызывать больше вопросов. При генерации мы хотим получить результат, который будет тяжело отличить от работы реальных художников или архитекторов, однако, чтобы оценить это сходство нет четких критериев и, в основном, приходится полагаться на субъективные ощущения пользователей при демонстрации конечного продукта.[1]

Исходя из вышесказанного, целью этого исследования будет является достижение более реалистичных результатов при генерации жилых помещений.

Задачей данного исследования является изучение существующих подходов генерации жилых помещений и описание критериев для оценки реалистичности генерации исходя из правил эргономики и эффективности использования пространства.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Существующие подходы

Для приложений, в которых используется компьютерная графика требуются модели, которые обычно создаются вручную с помощью специализированного программного обеспечения для 3D моделирования. В архитектуре они требуются для создания 3D-зданий из чертежа или планировки для предварительной визуализации интерьера или экстерьера недостроенных зданий. В видеоиграх они используется для создания игровых уровней. Задачи их создания нетривиальны, требуют умения и времени.

Однако существует несколько решений для снижения затрат времени на моделирование. В архитектурных приложениях существуют автоматизированные системы, которые преобразуют 2D-чертежи планировок в 3D модели [31]. В видеоиграх процедурные алгоритмы могут использоваться для создания контента [7], включая виртуальные пространства, такие как города [8,19,28,29], и природные ландшафты. Процедурная генерация применяется и при создании моделей обычных объектов: мебель [4,14], здания [5,16,17,32], персонажи.

Также было представлено несколько методов генерации 3D-модели интерьера здания с автоматическим созданием планировки. Одна из причин процедурного создания планировок для конкретных форм зданий заключаются в том, что процедурные генераторы обычно создают полые оболочки зданий без 3D-модели интерьера. Вторая причина — планирование и оптимизация реальных архитектурных планировок [2, 15, 24, 30].

В последнее время можно выделить четыре основные категории процедурных методов генерации планировки. Первый тип берет границы здания и делит внутреннее пространство, чтобы создать комнаты. Примеры этого типа метода включают работы Ноэля и Шивон [18], Хана и др. [6], Ринди и др. [23], Марсона и др. [11] и Лиу и др. [9]. Второй тип принимает обратный подход, создавая здания изнутри наружу. Помещения расположены в соответствии с количеством смежных комнат, и не используют границы

здания в качестве ограничения. Примером этого являются методы, представленные Мартином [12] и Меррелл и др. [13]. Третий тип метода создания планировки распределяет прототипы комнат в границах здания и итеративно увеличивает комнаты до их окончательного размера, занимая все внутреннее пространство. Примеры этих методов, основанных на росте, представлены Тьютнелом и др. [26] и Лопес и др. [10]. Последний тип представлен в работе Пэнг и др. [20], который разбивает область на четырехугольную сетку и полностью покрывает область плиткой, которые в этом случае представляют комнаты.

У каждого подхода есть преимущества и недостатки. Метод подразделения эффективный и простой, но он не обеспечивает достаточного контроля над формой комнаты и ее соединениями с другими объектами. Полученные планировки либо игнорируют типы комнат и соединения [23] или подходят только для правильных форм контура [9,11]. Второй подход, состоящий из создания здания изнутри наружу, может использоваться для создания реалистичных планов этажей, но имеет ограниченное использование, так как его нельзя применять к зданиям с уже заданным внешним видом. Подход размещения плитки, представленный Пенг и др. [20] может создать планировку для произвольного внешнего вида здания, покрывая доступное пространство сеткой из комнат, но не обеспечивает контроль над тем, как и где комнаты будут размещены. Наконец, основанные на росте алгоритмы позволяют избежать дорогостоящих шагов стохастических оптимизаций, но существуют проблемы, которые могут возникнуть из-за плохого размещения начальных комнат. Например, две комнаты, которые должны быть смежными, могут быть разделены из-за роста третьей комнаты. Кроме того, комната может не вырасти до приемлемого размера, если другие помещения, окружающие ее, мешают росту.

Для целей данного исследования было принято решение сосредоточить обзор на методах, которые генерируют планировочные решения внутри зданий.

1.2 Алгоритмы разбиения (Subdivision)

Изначально метод разбиения был представлен Хан и др. [6] для генерации планировки зданий в режиме реального времени. Их исследования сосредоточены на создании комнат и этажей на основе данных о положении пользователя, и удалении из памяти комнат, которые не находятся в зоне видимости. Это позволяет избежать ненужного использования памяти и вычислительных ресурсов, что позволяет процессу процедурной генерации обрабатывать большие здания с множеством разных комнат (например небоскребы).

Процесс также обеспечивает неизменную геометрию. Удаленная комната может быть восстановлена в прежнем виде путем повторного использования того же сида, который изначально использовался при ее создании, с последующим повторным применением всех изменений, которые пользователь мог внести. Когда регион удаляется из памяти, все изменения, сделанные в нем, сохраняются в хэш-карте.

В ходе этого процесса создаются планировки в виде сетки (см. рис. 1.1) путем случайного разделения плоскостями, которые выровнены по осям. Одиннадцать правил служат ограничениями, позволяющими избежать несоответствий в системе генерации и создавать реалистичные интерьеры (например, одно правило гарантирует, что все комнаты могут быть доступны из входа в здание).

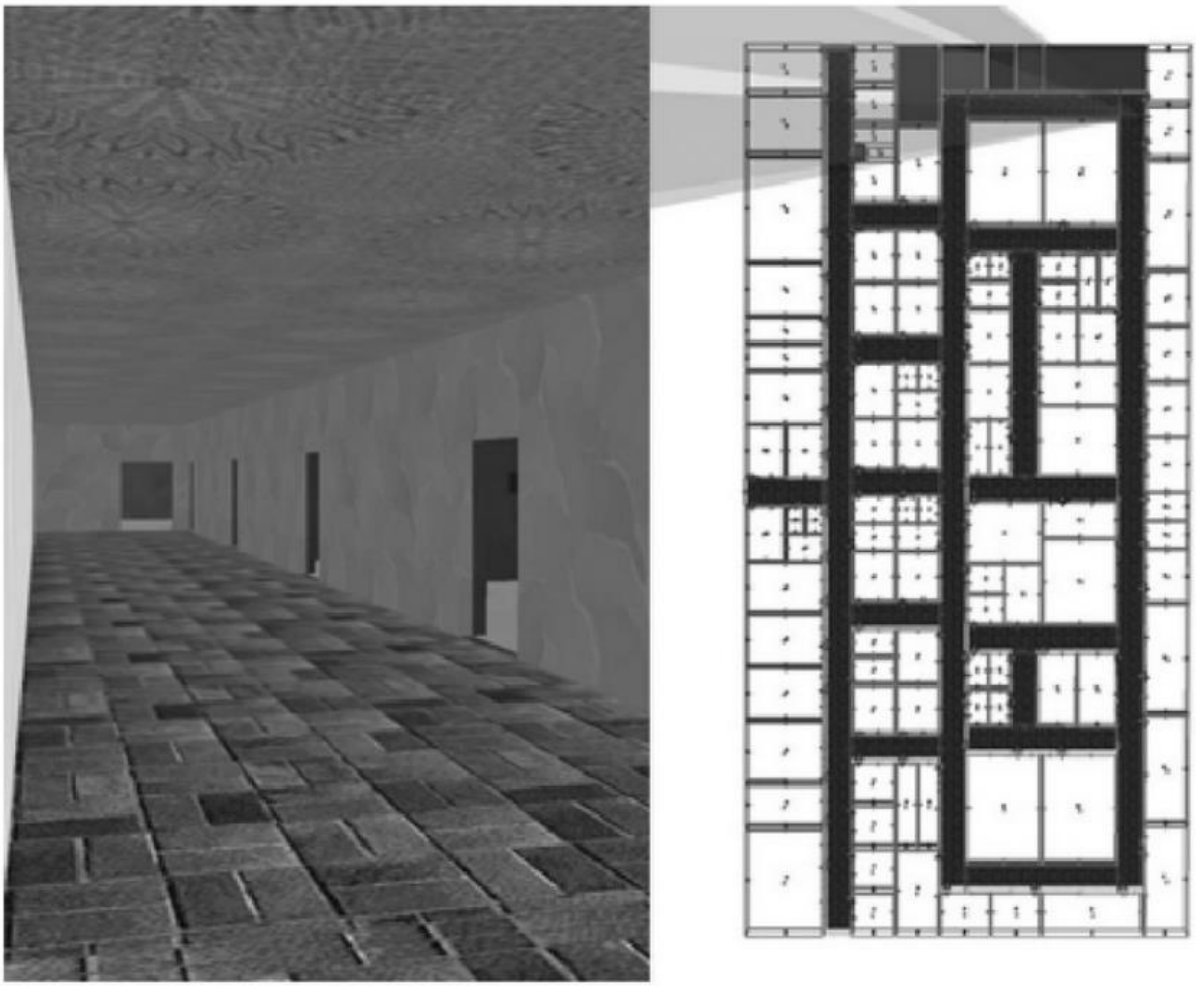

Рисунок 1.1: Снимок экрана, показывающий вид здания от первого лица и вид сверху на полностью сгенерированный пол. [6]

Здания, созданные с помощью этого метода, разделены на прямоугольные области, соединенные «порталами». Начиная с контура здания, алгоритм сначала размещает все элементы, охватывающие несколько этажей, такие как шахты лифтов и лестницы. Затем алгоритм разбивает здание на ряд этажей. После, каждый этаж разделяется коридором, который может быть прямым или прямоугольным. Наконец, внутренние помещения подразделяются на комнаты, и соответствующие предметы интерьера размещаются внутри.

Ринде и Даль [23] представляют метод, который выполняет подразделение экстерьера здания. Входными данными являются внешняя стена конструкции, а также положение и размеры любых окон и дверей. Также

есть и другие параметры, которые используются для контроля результата алгоритма различными способами, начиная от ширины коридора и заканчивая типами зон и комнат, которые должно содержать здание.

Во-первых, из входных данных создается каркас здания, чтобы облегчить анализ формы здания, а также конструкции внутренних зон, таких как коридоры. Это достигается с помощью модифицированной версии алгоритма прямого скелета [3], который создает скелет для многоугольника. В этом методе все наружные стены вдавливаются внутрь с постоянной скоростью, а скелет создается там, где встречаются две стены. Двери создаются из ребер, построенных перпендикулярно скелету (см. рис. 1.2 (a)). После того, как построен скелет, размещаются коридоры внутри здания. Эта транзитная зона используется для доступа ко всем частям интерьера. Создаются регионы, которые занимают наибольшее непрерывное пространство и используются для дальнейшего деления на субрегионы. Субрегионы представляют собой границы набора комнат (например, квартир см. рис. 1.2 (b)). Стены комнаты построены с использованием алгоритма гибридного псевдо-Вороного/S-пространства (см. рис. 1.2 (c)), где ячейка границы диаграммы Вороного используются для выбора подходящих ребер S-пространства для стен (см. рис. 1.1.(d)). Наконец, комнаты строятся на основе полученных границ. При этом строится график потенциальных соседних комнат, который затем используется для определения типов комнат и размещения дверей между ними. На рис. 1.2 (e) показан план этажа, созданный из наружных стен.

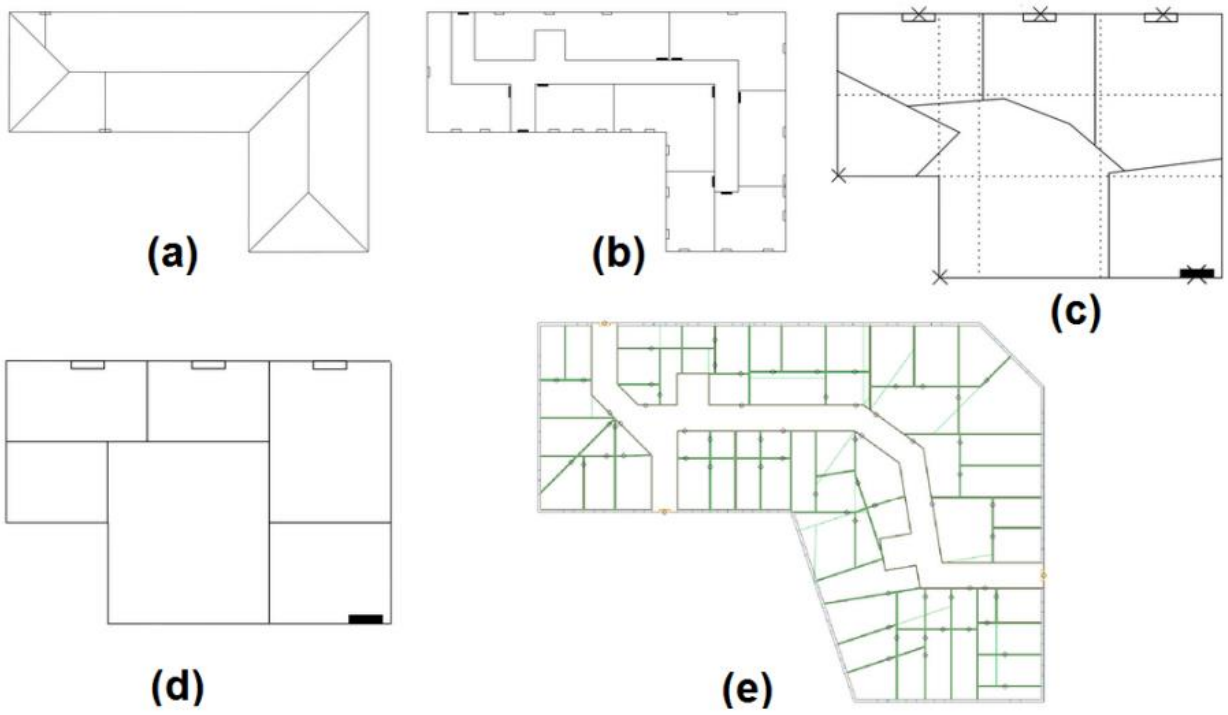

Рисунок 1.2: Прямой скелет со входными дверьми (a), деление на регионы и далее на субрегионы для создания коридоров и квартир (b), псевдо-Вороной/S-Space алгоритм для выбора ребер для стен (c), готовый план квартиры (d) план этажа, созданный с помощью этого метода. (e). [23]

Марсон и Мюсс [11] представили метод, который подразделяет доступную площадь, используя алгоритм квадратных древовидных карт (см. Брулс и др. [1]), такой, что процесс подразделения пытается поддерживать каждую комнату с соотношением сторон, равным единице. Входными данными для алгоритма являются размеры здания, желаемая ширина и длина для каждой комнаты, а также тип для каждой комнаты, который определяет возможные связи между ними.

На первом этапе процесса применяется алгоритм квадратной древовидной карты для разделения здания на три зоны: общественную, частную и обслуживающую. На втором этапе снова применяется квадратная древовидная карта для каждой зоны, создавая комнаты. После создания комнат соединения между комнатами размещаются в соответствии с деревом связей. Последним шагом является проверка того, что все комнаты можно пройти от входа. Если

доступ к комнате невозможен (см. рис. 1.3), процесс создает коридор (см. рис. 1.4).

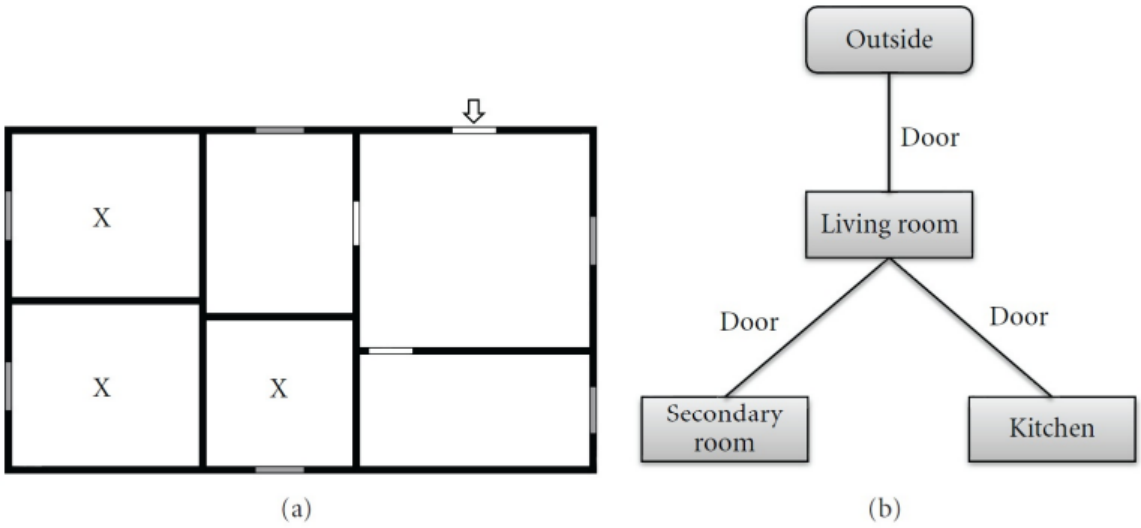

Рисунок 1.3: Пример плана этажа (a), содержащего 3 недоступные со входа комнаты (обозначены X) и их соответствующий граф связности (b). [11]

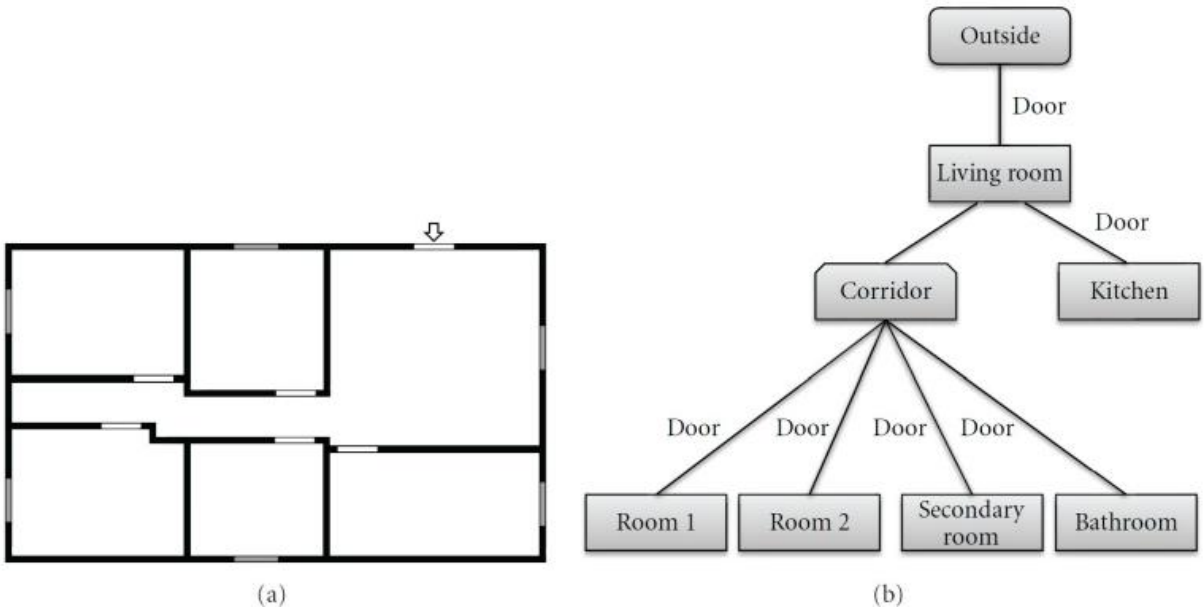

Рисунок 1.4: План этажа, показанный на Рисунке 1.3, изменен, чтобы включить коридор (a) и соответствующий ему граф связности (b). [11] 1.3

1.3 Алгоритмы размещения сетки (Tile placement)

Пэн и др. [20] описали метод разбиения области на множество изменяемых шаблонов, таких что пространство может быть полностью ими покрыто без пересечения шаблонов. Метод состоит из двух этапов: на первом определяется приблизительные позиции шаблонов, на втором задаются их формы. На первом этапе пространство и шаблоны разбиваются на четырехугольные сетки (см. рис. 1.5 (a), (b) и (c)). Ищется непрерывная сетка в основной области для встраивания шаблонов (см. рис. 1.5 (d)). Так как форма размещенной сетки может отклоняются от исходного шаблона, сетка оптимизируются непрерывной квадратичной оптимизацией до сходимости или до тех пор, пока не будет достигнут предел времени (см. рис. 1.5 (e)).

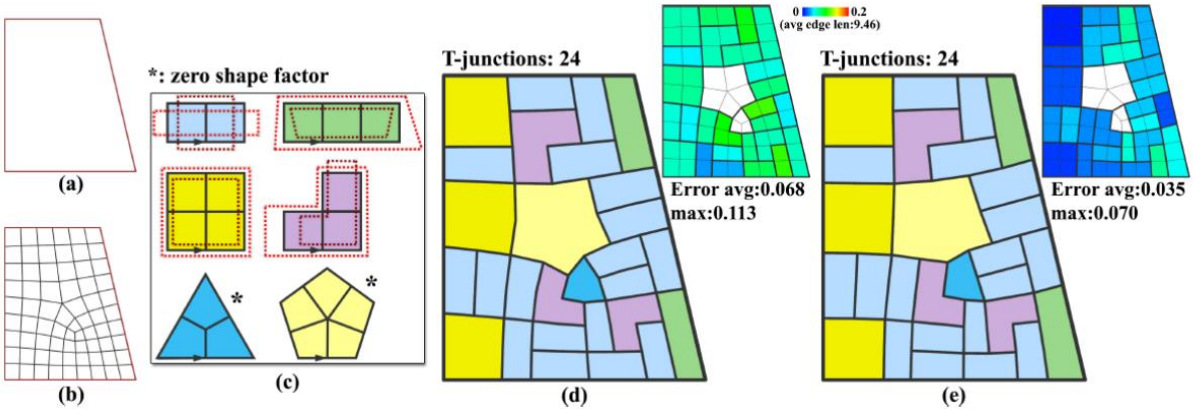

Рисунок 1.5: входная область (a) и ее разбиение на четырехугольники (b). шаблоны комнат (c). Каждый шаблон показан в виде его базового полигона и его допустимых преобразований. Полная сгенерированная сетка (d). Погрешность при изменении формы визуализируются в правые верхние углы. Геометрия сетки оптимизирована для дальнейшего уменьшения погрешности. (e). [20]

Этот метод подходит для решения широкого круга задач планировки, в том числе для создания планировок крупных объектов, таких как офисы, торговые центры и больницы. Во время размещения сетки, дополнительные топологические ограничения используются для обеспечения размещения

коридора и доступности комнат. Критерии доступности выполняются за счет использования коридора, соединяющих комнаты со входами в здание. Изменение шаблонов комнат позволяет пользователю добиться желаемого внешнего вида. Шаблоны коридоров реализуются при помощи многоуровневого ветвящегося дерева. На рис. 1.6 показаны две разные планировки офисного здания, полученные в результате изменения изначальных положений коридорных узлов.

Рисунок 1.6: Планы двух офисных этажей. Слева: шаблоны коридора и комнаты, использованные при размещении. Справа: два разных плана этажей сгенерированных из различных начальных положений корневых узлов коридора. [20]

1.4 Алгоритмы «наизнанку» (Inside out)

Мартин [12] описал алгоритм для создания поэтажных планов жилых зданий (см. рис. 1.7). Процесс состоит из трех основных этапов.

На первом этапе базовая структура дома представляется графом, в котором каждый узел соответствует комнате, а каждая связь соответствует проходу между комнатами. Сам график генерируется путем использования входной двери в качестве корневого узла с последующим добавлением всех комнат общего пользования и, наконец, добавление всех персональных комнат. Конкретные типы комнат изначально не назначаются, а определяются после генерации общественных комнат в соответствии с набором пользовательских атрибутов.

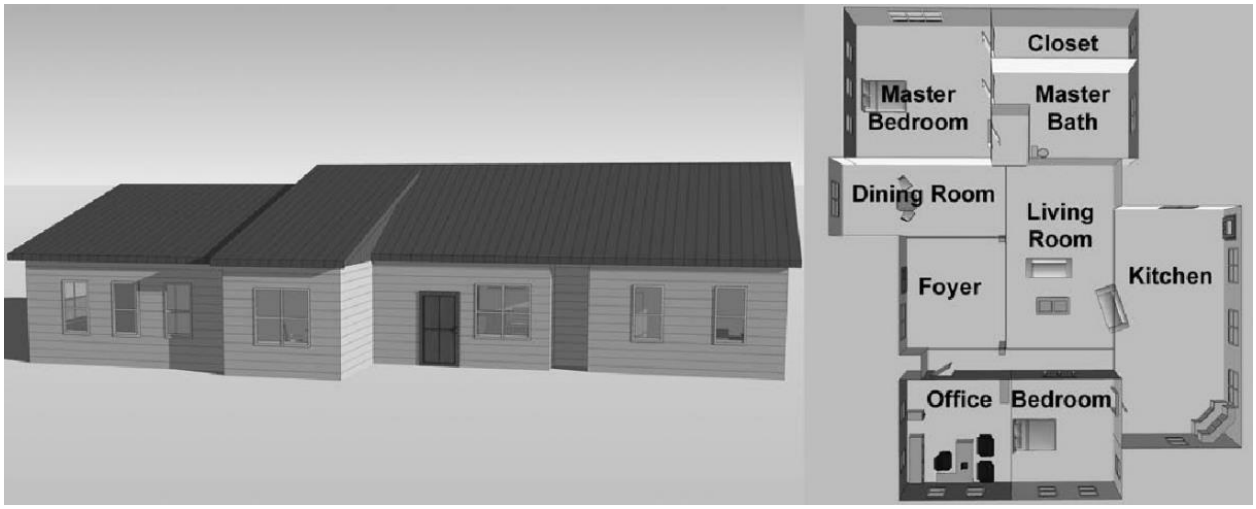

Рисунок 1.7: слева: смоделированный дом на основе плана этажа, созданного с помощью алгоритма Мартина. Справа: Вид сверху на план того же дома. [12]

В начале второго этапа каждому помещению был присвоен тип и связи между комнатами определены, но комнаты еще не расположены в пространстве. Второй этап генерирует 2D положение каждой комнаты из графика, созданного на первом этапе. Для этого процесс рассматривает граф как дерево с корнем в комнате, примыкающей к входной двери. Из этого корня каждый дочерний узел равномерно распределяется по соседству, на равном

расстоянии друг от друга и от корневой комнаты. Затем процесс повторяется для каждого из дочерних узлов.

На третьем этапе комнаты расширяются до их окончательного размера. Каждая комната оказывает внешнее "давление", которое пропорционально ее размеру. Таким образом, каждая комната расширяется, чтобы занять оставшееся пространство здания. Если стена является общей для двух комнат, алгоритм учитывает давление внутри и снаружи, чтобы определить, будет ли комната увеличена.

Меррелл и др. [13] представляют метод создания реалистичных планировок жилых зданий из спецификации пользователя. По их мнению, критерии, используемые архитекторами, слишком многочисленны и плохо определены для моделирования с помощью системы, основанной на четких правилах. Так что вместо этого они применяют техники машинного обучения. Байесовская сеть обучается на данных реальных зданий, чтобы выводить аспекты архитектурного дизайна, которые являются субъективными и трудно кодифицируемыми. Такие данные включают, например, какие типы комнат часто являются смежными, площадь комнат и соотношение сторон.

На первом этапе процесса процедурной генерации определяется разнообразный набор высокоуровневых требований. Специфицируются требования, такие как количество спален или приблизительная площадь. Затем Байесовская сеть используется для перевода требований в архитектурный план с описанием смежных комнат, желаемой площадью и соотношением сторон для каждой комнаты.

Второй этап превращает архитектурный план в детальную планировку каждого этажа здания. Это достигается за счет стохастической оптимизации пространства возможного построения здания [22].

На каждой итерации алгоритма создается новая планировка здания с локальными и глобальными реконфигурациями, которые существенно изменяют общую компоновку здания. Реконфигурация планировки достигается за счет двух действий: движение стен и перестановка комнат.

Функция затрат, которая оценивает качество предлагаемой планировки учитывает доступность комнат, площадь, соотношение сторон и форму каждой комнаты. На рис. 1.8 представлена планировка, оптимизированная под заданный архитектурный план.

Рисунок 1.8: Компоновка здания, сгенерированная компьютером. Слева: архитектурный план, созданный Байесовской сетью, обученной на реальных данных. Справа: набор планировок, оптимизированных для архитектурного плана. [13]

1.5 Алгоритмы расширения (Growth-based)

Тьютнел и др. [26] предлагают метод построения планировки здания на основе алгоритма расширения. Каждый отдельный тип комнаты определяется как класс в семантической библиотеке, и для каждого класса может быть установлено несколько отношений. Отношения определяют, какие проходы между комнатами разрешены, но также могут определить дополнительные ограничения, например требование, чтобы гараж располагался дверью к улице.

Для размещения комнат выполняются два шага. Во-первых, прямоугольники, называемые характерными областями, помещаются в места, где все классовые отношения удовлетворены. Затем все комнаты итеративно расширяются, пока не коснутся друг друга.

На рис. 1.9 представлены три планировки здания, созданные с помощью характерных областей, размещенных по-разному внутри здания фиксированной формы. Для размещения характерных областей соблюдаются определенные правила. Например, в этом случае прихожая должна касаться наружной стены со стороны улицы, а гостиную должна находиться рядом с кухней.

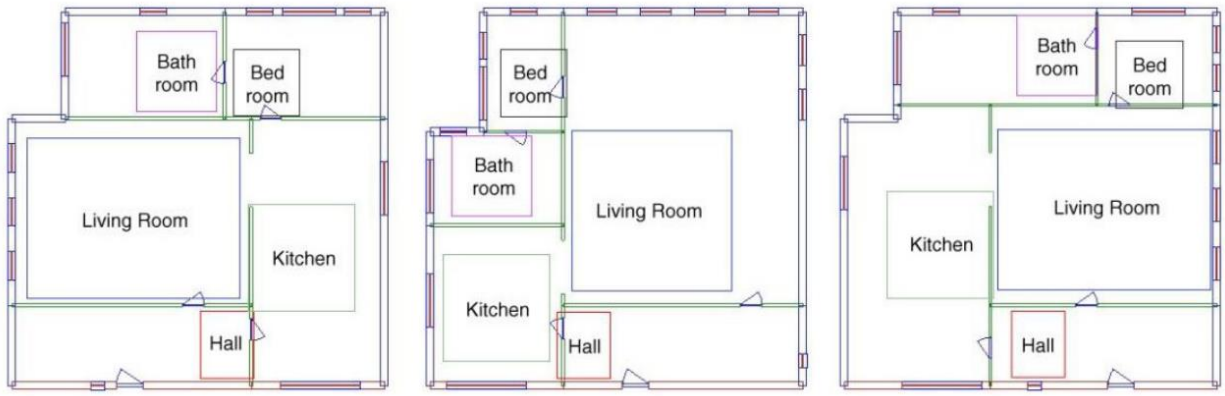

Рисунок 1.9: Три разных примера планировки, созданные путем изменения положения характерных областей. [26]

Этот алгоритм также использует семантическую библиотеку для определения взаимосвязей и ограничений, которые служат ориентиром для распределения объектов (например, мебели) в комнате. Все возможные места размещения определяются с учетом требований конкретного объекта, таких как требуемый зазор вокруг предмета или необходимость стоять у стены.

Лопес и др. [10] отмечают, что другие методы не могут обрабатывать помещения неправильной формы, например, Г-образные или П-образные комнаты. Они предлагают алгоритм расширения на основе сетки, который увеличивает комнаты в пределах планировки здания.

Входными данными для процесса являются размер ячейки сетки в метрах, список комнат для генерации, ограничения смежности, а также для каждой комнаты свой тип (общественная, частная или проходная) и предпочтительный размер.

Для создания планировки выполняются следующие шаги. Во-первых, алгоритм делит пространство, создавая частные и общественные зоны.

Во-вторых, выполняется этап размещения комнат. Для этого процесса используется отдельная сетка с весом каждой ячейки. Изначально каждая ячейка снаружи здания получает нулевой вес, а каждая ячейка внутри получает вес единицы. Веса в каждой ячейке затем изменяются по мере необходимости в соответствии с ограничениями. Например, чтобы гарантировать, что комната касается внешней части здания, ячейки, которые не касаются улицы, получают нулевой вес. На основе этих весов сетки одна ячейка выбирается для размещения комнаты.

Третий шаг расширяет комнаты. Алгоритм выбирает одну комнату за раз и пытается вырастить ее на один шаг. Помещения, требующие большей площади, выбираются чаще и расширяются быстрее. Когда все комнаты расширены до максимальной прямоугольной формы, процесс пытается расширить комнаты в L-образной форме. На последнем этапе алгоритм сканирует сетку на наличие пустых ячеек. и назначает их в соседнюю комнату,

заполняя пробелы. На рис. 1.10 показан пример планировки, сгенерированный данным методом.

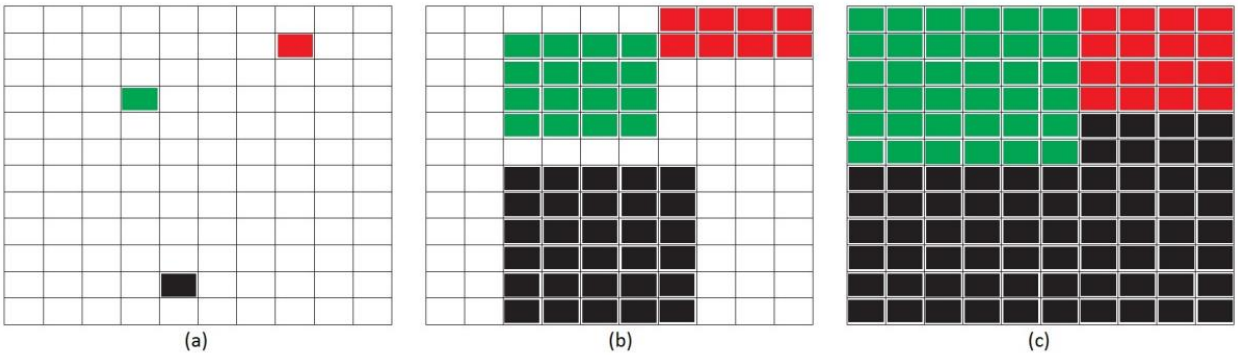

Рисунок 1.10: Пример алгоритма расширения: (a) начальные положения комнаты, (b) прямоугольная рост, показывающий шаг, на котором комнаты достигли своего максимального размера, (c) L-образный рост. [10]

1.6 Выявленные проблемы

Приведенные подходы отличаются между собой задаваемыми пользовательскими ограничениями и временем выполнения. Однако Главной задачей этих методов является правдоподобная генерация помещений. И при оценке этой правдоподобности могут возникать проблемы. Чаще всего достижение правдоподобности достигается все большим количеством накладываемых ограничений на объекты, но при этом не учитывается взаимное положение этих объектов относительно друг друга.

Одним из вариантов решения данной проблемы может выступать оценка эргономичности помещения и эффективности использования пространства. Для получения этой оценки можно использовать автоматическую программу, которая будет симулировать рутинную деятельность человека в условиях заданного помещения (например: лечь на диван и включить телевизор, подойти к плите и приготовить еду, снять одежду и лечь в кровать). На основе выполненных действий программа будет выдавать оценку эффективности на основе множества параметров в т.ч. время на выполнение операций, расстояние, пройдённое во время выполнения, количество объектов, мешавших достижению цели и т.д. Для описания системы оценки возможно использование сформированных правил эргономичного оформления жилых пространств [34]. На рисунке 1.11 показано, как расстановка мебели в комнате не должна находится на пути типичных маршрутов жильцов. На рисунке 1.12 показаны различные конфигурации кухонного пространства, где виден маршрут человека во время готовки, ограниченный ключевыми объектами: холодильник, раковина, плита.

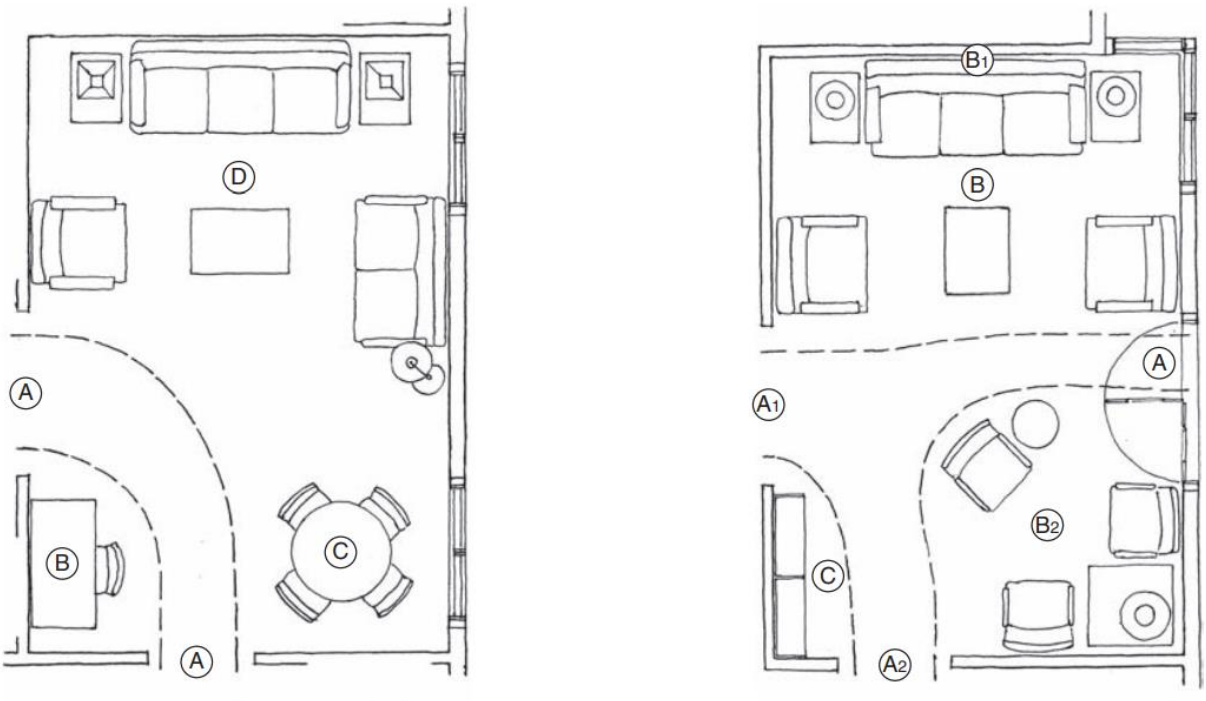

Рисунок 1.11: планировка помещения, оптимизированная с учетом эффективного использования пространства.

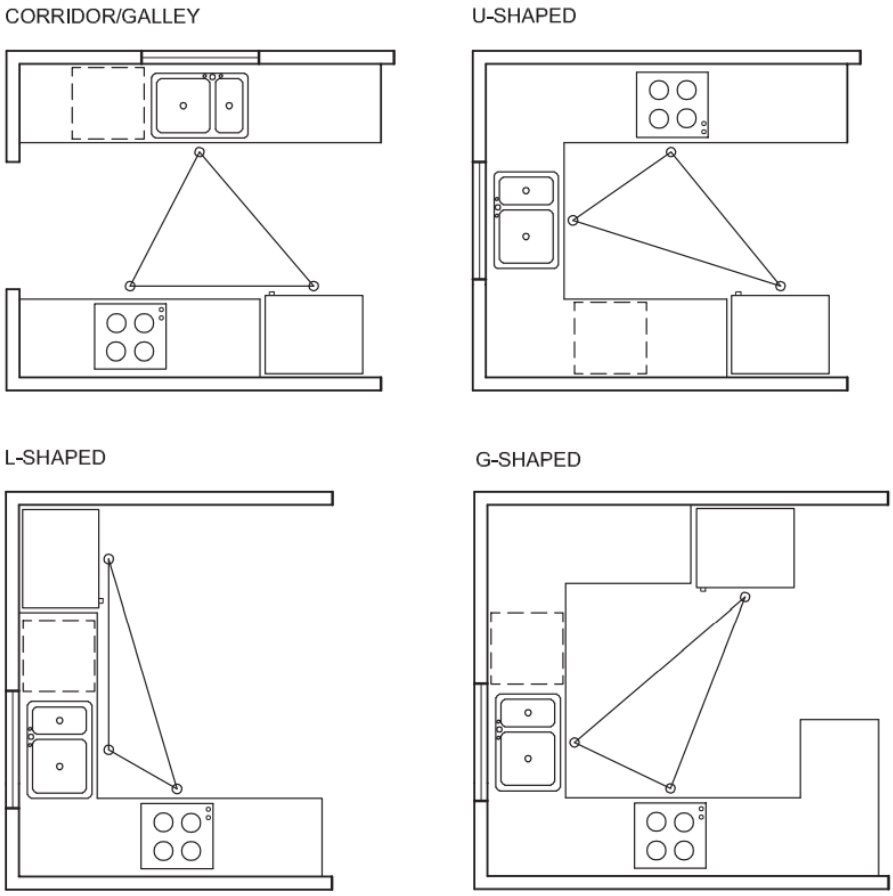

Рисунок 1.12: планировка кухонного пространства с демонстрацией пути перемещения человека между ключевыми точками

2 МЕТОДОЛОГИЯ

2.1 Методы генерации

Для приложений, в которых используется компьютерная графика требуются модели, которые обычно создаются вручную с помощью специализированного программного обеспечения для 3D моделирования. В архитектуре они требуются для создания 3D-зданий из чертежа или планировки для предварительной визуализации интерьера или экстерьера недостроенных зданий. В видеоиграх они используется для создания игровых уровней. Задачи их создания нетривиальны, требуют умения и времени.

Однако существует несколько решений для снижения затрат времени на моделирование. В архитектурных приложениях существуют автоматизированные системы, которые преобразуют 2D-чертежи планировок в 3D модели [31]. В видеоиграх процедурные алгоритмы могут использоваться для создания контента [7], включая виртуальные пространства, такие как города [8,19,28,29], и природные ландшафты. Процедурная генерация применяется и при создании моделей обычных объектов: мебель [4,14], здания [5,16,17,32], персонажи.

Также было представлено несколько методов генерации 3D-модели интерьера здания с автоматическим созданием планировки. Одна из причин процедурного создания планировок для конкретных форм зданий заключаются в том, что процедурные генераторы обычно создают полые оболочки зданий без 3D-модели интерьера. Вторая причина — планирование и оптимизация реальных архитектурных планировок [2, 15, 24, 30].

В последнее время можно выделить четыре основные категории процедурных методов генерации планировки. Первый тип берет границы здания и делит внутреннее пространство, чтобы создать комнаты. Примеры этого типа метода включают работы Ноэля и Шивон [18], Хана и др. [6], Ринди и др. [23], Марсона и др. [11] и Лиу и др. [9]. Второй тип принимает обратный подход, создавая здания изнутри наружу. Помещения расположены в

соответствии с количеством смежных комнат, и не используют границы здания в качестве ограничения. Примером этого являются методы, представленные Мартином [12] и Меррелл и др. [13]. Третий тип метода создания планировки распределяет прототипы комнат в границах здания и итеративно увеличивает комнаты до их окончательного размера, занимая все внутреннее пространство. Примеры этих методов, основанных на росте, представлены Тьютнелом и др. [26] и Лопес и др. [10]. Последний тип представлен в работе Пэнг и др. [20], который разбивает область на четырехугольную сетку и полностью покрывает область плиткой, которые в этом случае представляют комнаты.

У каждого подхода есть преимущества и недостатки. Метод подразделения эффективный и простой, но он не обеспечивает достаточного контроля над формой комнаты и ее соединениями с другими объектами. Полученные планировки либо игнорируют типы комнат и соединения [23] или подходят только для правильных форм контура [9,11]. Второй подход, состоящий из создания здания изнутри наружу, может использоваться для создания реалистичных планов этажей, но имеет ограниченное использование, так как его нельзя применять к зданиям с уже заданным внешним видом. Подход размещения плитки, представленный Пенг и др. [20] может создать планировку для произвольного внешнего вида здания, покрывая доступное пространство сеткой из комнат, но не обеспечивает контроль над тем, как и где комнаты будут размещены. Наконец, основанные на росте алгоритмы позволяют избежать дорогостоящих шагов стохастических оптимизаций, но существуют проблемы, которые могут возникнуть из-за плохого размещения начальных комнат. Например, две комнаты, которые должны быть смежными, могут быть разделены из-за роста третьей комнаты. Кроме того, комната может не вырасти до приемлемого размера, если другие помещения, окружающие ее, мешают росту.

2.2 Алгоритмы генерации пространств

Для генерации жилых помещений нам необходимо выбрать алгоритм разбиения периметра здания на комнаты. Этот алгоритм должен получать на вход векторное представление контура помещения и список комнат, необходимых для размещения. После этого комнаты должны быть размещены внутри пространства с соблюдением индивидуальных ограничений, таких как: иметь доступ друг к другу, быть соединены дверьми, иметь окна, не пересекаться и т.д.

После проведения анализа предметной области были выявлены различные методы разбиения пространства, которые будут представлены ниже. Названия основных методов:

1) Subdivision
2) Inside out
3) Tile placement
4) Growth-based

Каждый метод обладает своими плюсами и минусами. В результате предыдущей работы было решено использовать метод Growth-based, так как он больше всего советует заявленным критериям и при небольших модификациях этого метода возможно добиться желаемых результатов.

Входными данными для процесса являются размер ячейки сетки в метрах, список комнат для генерации, ограничения смежности, а также для каждой комнаты свой тип (общественная, частная или проходная) и предпочтительный размер.

Для создания планировки выполняются следующие шаги. Во-первых, алгоритм делит пространство, создавая частные и общественные зоны.

Во-вторых, выполняется этап размещения комнат. Для этого процесса используется отдельная сетка с весом каждой ячейки. Изначально каждая ячейка снаружи здания получает нулевой вес, а каждая ячейка внутри получает вес единицы. Веса в каждой ячейке затем изменяются по мере необходимости в соответствии с ограничениями. Например, чтобы

гарантировать, что комната касается внешней части здания, ячейки, которые не касаются улицы, получают нулевой вес. На основе этих весов сетки одна ячейка выбирается для размещения комнаты.

Третий шаг расширяет комнаты. Алгоритм выбирает одну комнату за раз и пытается вырастить ее на один шаг. Помещения, требующие большей площади, выбираются чаще и расширяются быстрее. Когда все комнаты расширены до максимальной прямоугольной формы, процесс пытается расширить комнаты в L-образной форме. На последнем этапе алгоритм сканирует сетку на наличие пустых ячеек. и назначает их в соседнюю комнату, заполняя пробелы. На рис. 1.1 показан пример планировки, сгенерированный данным методом.

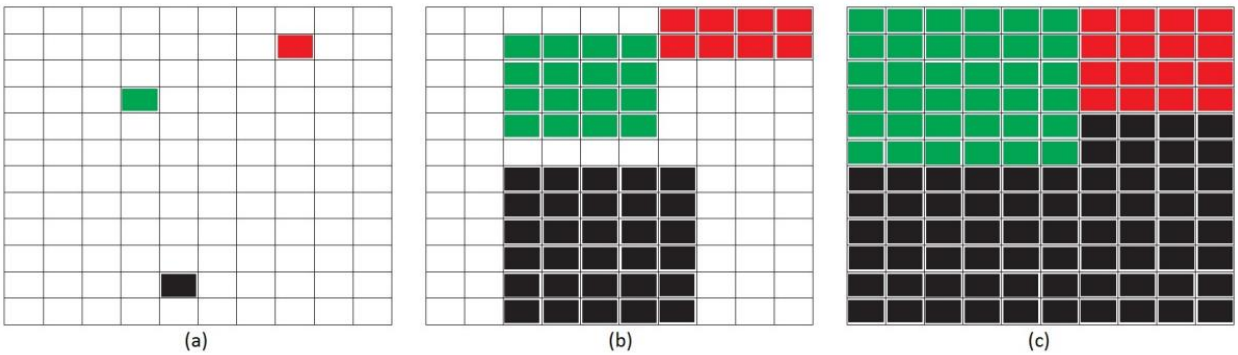

Рисунок 1.1: Пример алгоритма расширения: (a) начальные положения комнаты, (b) прямоугольная рост, показывающий шаг, на котором комнаты достигли своего максимального размера, (c) L-образный рост. [10]

2.3 Алгоритм генерации расстановки мебели

После того, как будет сгенерирован контур помещения и комнат, нам необходимо сгенерировать расстановку мебели. Определенный набор типов мебели привязан к каждой комнате и задается вручную пользователем из заранее подготовленных моделей. Вариации мебели можно расширять, но для каждого объекта необходимо указывать соответствующие типы ограничений, например: занимаемое пространство, зона видимости, иерархические отношения, попарные отношения.

Оптимизация расстановки мебели для создания реалистичной и функциональной планировки требует множества ограничений, учитывая различные взаимодействующие факторы, такие как попарные отношения мебели, пространственные отношения по отношению к комнате и другие человеческие факторы. Необходимо эффективное представление, отражающее пространственные отношения.

2.3.1 Параметры объектов

Ограничивающие поверхности: каждый объект в сцене представлен набором ограничивающих поверхностей (это может быть простая прямоугольная рамка или выпуклая оболочка для решения более сложных пространственных задач). На рис. 1.2 показан пример объекта (диван), представленного ограничивающей рамкой, шесть поверхностей которой помечены цифрами от 1 до 6. Помимо верхней и нижние поверхности, мы ищем «заднюю» поверхность каждого объекта, которая является поверхностью, ближайшей к любой стене. Другие поверхности помечены как «незадние» поверхности. Задняя поверхность используется для определения базовой плоскости для назначения других атрибутов.

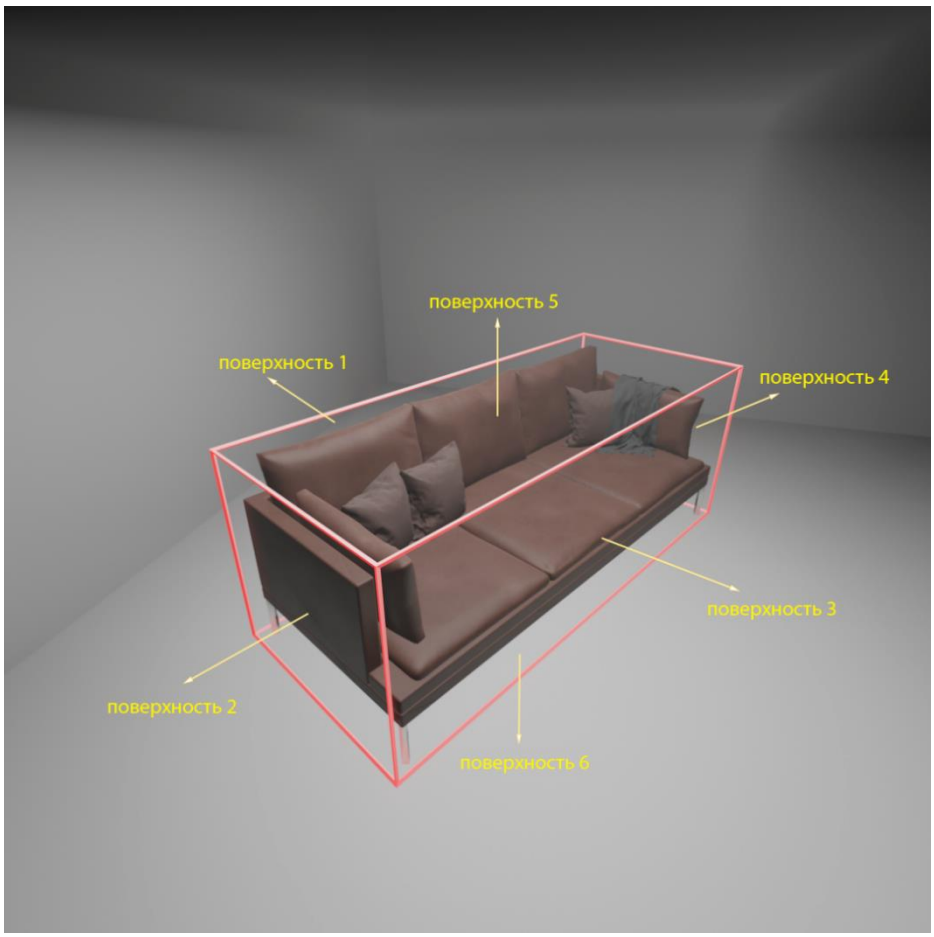

Рисунок 1.2: Диван, его ограничивающее пространство и 6 поверхностей

Центр и ориентация. На рис. 1.3 показаны основные атрибуты объект — центр и ориентация, обозначаемая (pi, θi), где pi обозначает координаты (x, y), а θi — угол относительно ближайшей стены (определяемый как угол между ближайшей стеной и задней поверхностью). Оптимизированное расположение мебели {(pi, θi)} с участием всех объектов i — это то, которое минимизирует функцию стоимости, определенную в следующем разделе.

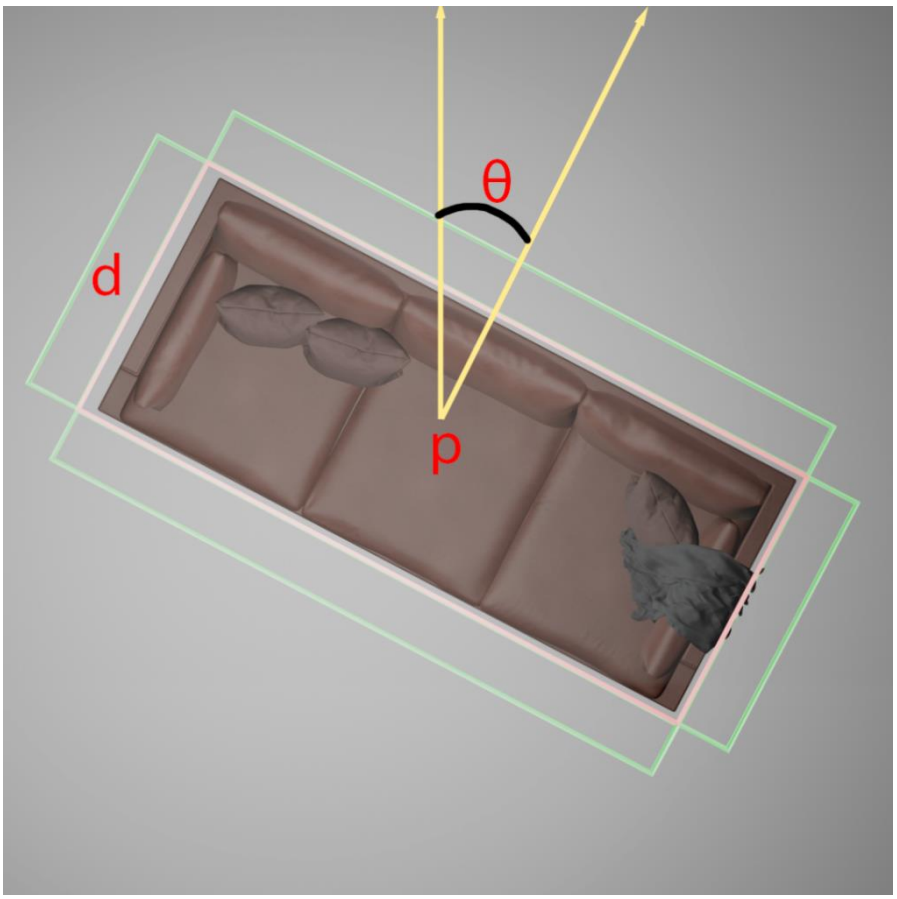

Рисунок 1.3: пример объекта, показывает угол к стене, индекс, ширину сечения, доступное пространство

Доступное пространство: для каждой поверхности объекта мы назначаем соответствующее доступное пространство (см. рис. 1.3). Мы определяем a_{ik} как координаты центра доступного пространства k объекта i. Диагональ региона измеряется ad_{ik}, который используется для измерения того, как глубоко другие объекты могут проникать в пространство во время оптимизации. Размер доступного пространства задается при определении параметров объекта, задающегося относительно размеров человеческого тела.

Зона видимости: для некоторых объектов, таких как телевизор и картина, лицевая поверхность должна быть видна. Мы назначаем зону видимости к этой конкретной поверхности. Учитывая объект i, его зона видимости задается как усеченная пирамида и аппроксимируется серией прямоугольников с центральными координатами v_{ik}, где k — индекс

прямоугольника. vd_{ik}- это диагональ прямоугольник, которая нужна для определения стоимости пересечения, аналогичной стоимости для доступного пространства. На рис. 1.4 приведен пример.

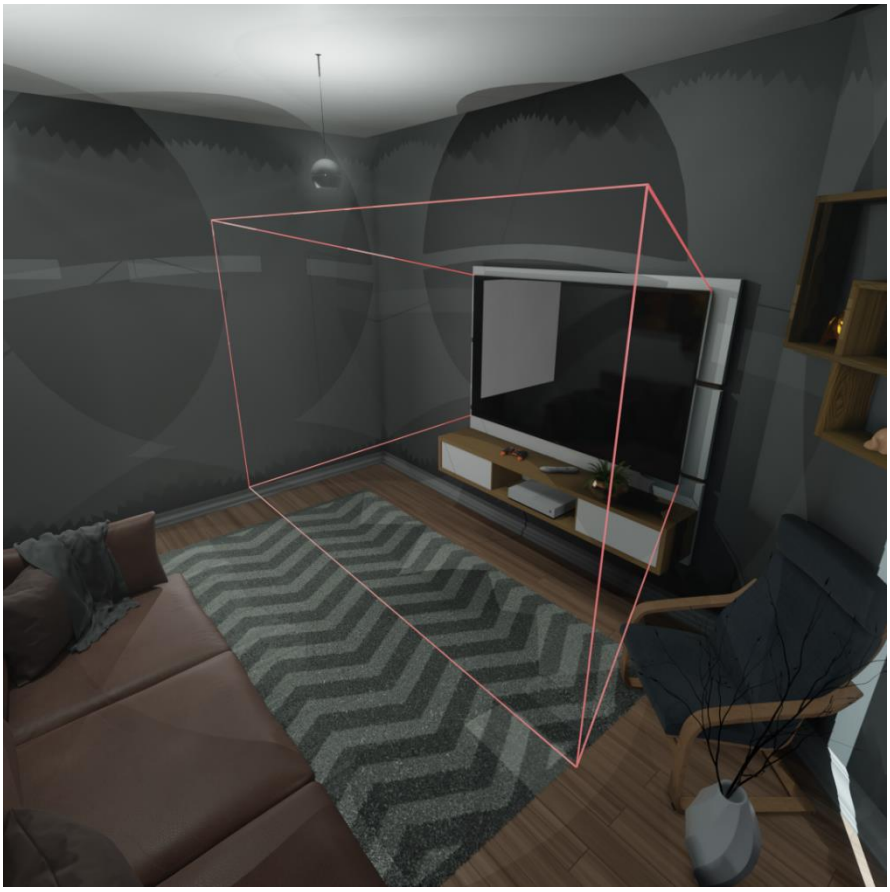

Рисунок 1.4: зона видимости объекта

Другие атрибуты: также в оптимизации учувствуют и другие атрибуты. Снова обращаясь к рисунку 1.3, расстояние от p_i до ближайшей стены определяется как d_i; диагональ от p_i до угла границы определена как b_i (текущая реализация представляет собой прямоугольник). Мы также записываем z-позицию z_i объекта.

Для упрощения процесса оптимизации перемещение объекта рассматривает только (x, y)-пространство. Другими словами, z-позиция объекта фиксируется как z-позиция поверхности его родителя первого уровня. Тем не менее, z-позиция все еще может измениться при перестановке. шаг,

когда объект второго уровня меняет своего родителя первого уровня и размещены на другой поверхности. Возможные столкновения в z-измерении будет по-прежнему учитываться при оценке стоимости доступности и видимости. Например, пересечение между стулом и кроватью в (x, y) пространстве штрафуется, а расстояние между настенными часами и кроватью нет, так как первое включает столкновение в z-измерении, а последнее нет. Таким образом, кресло стремится отойти от кровати в пространство (x, y), а настенные часы - нет.

2.3.2 Отношения объектов

Следующим типом ограничений являются отношения между объектами. Они могут быть разделены на представленные категории:

Пространственные отношения: ключевыми отношениями являются расстояние объекта к его ближайшей стенке d_i и его относительной ориентации к стене θ_i. Они соответственно задаются как входные параметры, где мы можем назначить каждому объекту соответствующие параметры.

Иерархические отношения: задаются как два объекта A и B, объект A определяется как родитель B (B как дочерний элемент A), если B зависит от A. На рис. 1.5 показан джойстик на столе. Следовательно, стол является родителем джойстика, а джойстик — дитя стола. Предположим, что дан пример комнаты, населенной предметами мебели. Поскольку сама комната считается корнем, все объекты, непосредственно поддерживаемые полом или стеной, определяются как «объекты первого порядка» (например, кровать, стол, часы на стене). Все объекты, поддерживаемые поверхностью объекта первого порядка (например, вазы на шкафу) определяются как «объекты второго порядка». Таким образом, конфигурация помещения представлена по иерархии отношений. Для простоты эта оптимизация рассматривает только первый и второй уровни, которые должны охватывать большинство объекты интереса.

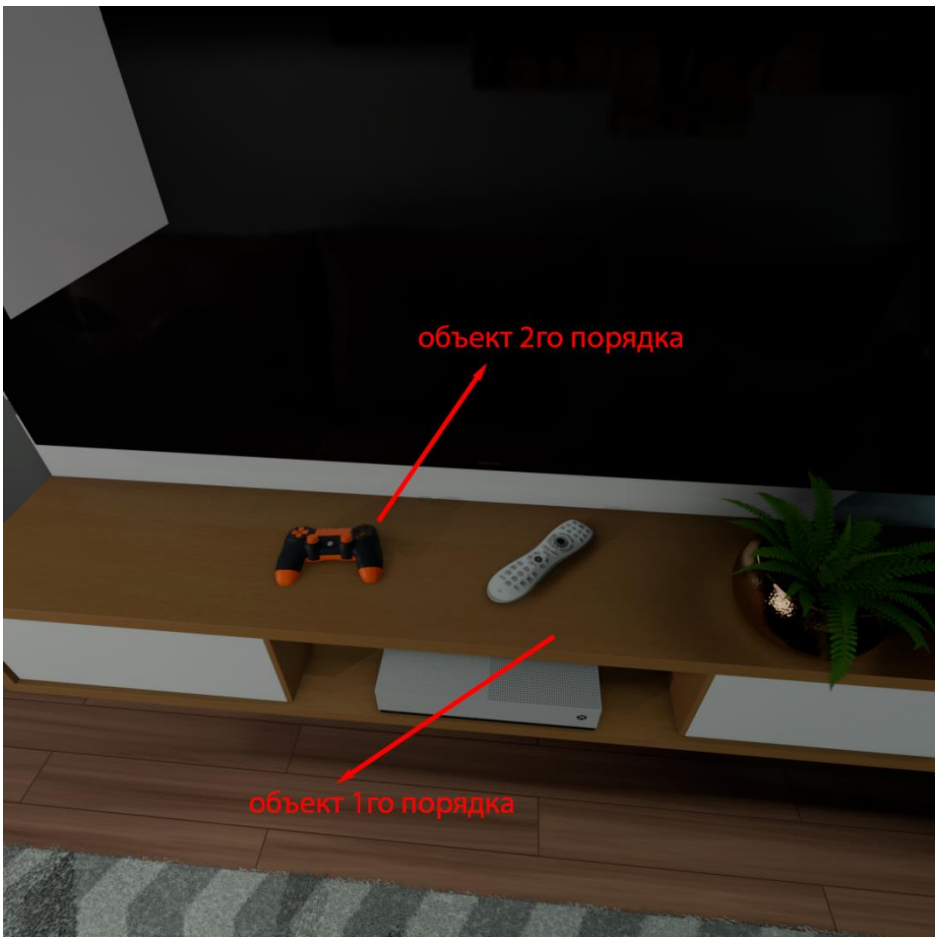

Рисунок 1.5: иерархические отношения объектов

Попарные отношения: определенные объекты, такие как телевизор и диван или обеденный стол и стулья, взаимодействующие друг с другом попарно с учетом попарной ориентации и ограничений расстояния. Каждое парное отношение можно установить у двух объектов при задании параметров, после чего указать среднее относительное расстояние и угол между ними.

2.3.3 Оптимизация размещения объектов

Учитывая пространственные отношения, как описано выше, наши цель состоит в том, чтобы интегрировать эту информацию в структуру оптимизации с правильно определенной функцией стоимости, количественно определяющей качество расстановка мебели. Учитывая произвольную планировку комнаты, заполненную предметами мебели, синтезированная

аранжировка должна быть полезна для моделирование виртуальной среды в играх, фильмах, дизайне интерьера, и других приложениях.

Искомое пространство в нашей задаче очень сложное, поскольку объекты взаимозависимы в процессе оптимизации. Позиции ориентации мебели зависят от множества факторов, таких как видимость или доступность. Очень трудно иметь глобальную схему оптимизации, которое дает единственный оптимум.

Чтобы решить эту проблему, можно прибегнуть к методам стохастической оптимизации, в частности, моделируемой закалке [34] с этапом поиска состояния Метрополис-Гастингс [37] для поиска хорошего приближения к глобальному оптимуму. Однако, при заданной комнате, наборе предметов мебели и предшествующих пространственных и иерархических отношениях возможны многочисленные приемлемые конфигурации. Это является основанием для нахождения хорошего результата за достаточно короткое время, вместо долгого поиска глобального оптимума функции стоимости.

Имитация закалки — это компьютерная имитация (физического) процесса закалки, при котором температура тепловой бани постепенно снижается, которая регулирует тепловую динамику твердого тела чтобы привести его в низкоэнергетическое равновесное состояние. Теоретически алгоритм гарантированно достигает глобального минимума с логарифмической скоростью при достаточно медленном графике охлаждения [36]. Использование такого медленного графика охлаждения, однако, непрактично. Тем не менее он широко использовался для поиска квазиоптимальных конфигураций в проектировании схем, операциях и многих других научных проблем. Как и в работе над созданием поэтажного плана [13], имитация закалки с простым критерием Метрополиса может быть эффективна в нашей задаче оптимизации конфигураций в пространстве возможных расстановок мебели.

По аналогии, предметы мебели в нашем приложении рассматриваются как атомы отжигаемого металла — они изначально «нагреты» чтобы обеспечить гибкую перестановку и уточнить их конфигурацию по мере температура постепенно снижается до нуля. При каждой температуре Критерий Метрополиса используется для определения вероятности перехода. Он использует целевую функцию типа Больцмана

$$f(\varphi) = e^{-\beta C(\varphi)}$$

где состояние системы $\varphi = \{(p_i, \theta_i) | i = 1, \ldots, n\}$ представляет собой конфигурацию мебели, состоящую из позиций p_i и ориентаций θ_i каждого из n предметов мебели, C — стоимость (энергия) функция, которая будет определена далее, а β обратно пропорциональна температуре, увеличиваясь на итерациях от высокой до низкой температуры. На каждой итерации новая конфигурация мебели φ’ , или «движение», предложено, и оно принимается с вероятностью

$$\begin{aligned}\alpha(\phi'|\phi) &= \min\left[\frac{f(\phi')}{f(\phi)}, 1\right] \\ &= \min\left[\exp(\beta(C(\phi) - C(\phi')), 1\right].\end{aligned}$$

Причем критерий Метрополиса может принимать ходы, увеличивающие стоимость, это позволяет методу избежать застревания на локальном уровне минимума.

2.3.4 Предложенные варианты перемещения

Чтобы эффективно исследовать пространство возможных расположений, предлагаемый ход $\varphi \rightarrow \varphi'$ включает в себя как локальную корректировку, которая изменяет текущую компоновку, так и глобальную реконфигурацию, которая меняет объекты местами, тем самым значительно меняя расположение.

Перевод и вращение: основной ход оптимизации изменяет положение объекта и его ориентацию. Для решения задачи расстановки мебели 2D-преобразования перемещения и вращения достаточны, чтобы придать объектам практически осуществимую форму расстановки, так как в большинстве случаев предметы мебели стоят вертикально на полу под действием силы тяжести. С математической точки зрения, объект i или подмножество объектов выбирается и обновляется при перемещении

$(p_i, \theta_i) \rightarrow (p_i+\delta p, \theta_i)$ или $(p_i, \theta_i) \rightarrow (p_i, \theta_i + \delta\theta)$,

где $\delta p \sim [N(0, \sigma_p^2)\ N(0, \sigma_p^2)]^T$ и $\delta\theta \sim N(0, \sigma_\theta^2)$,

где $(\mu, \sigma^2) = (\mu, \sigma^2) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{-(x-\mu)^2/2\sigma^2}$

нормальное (гауссовское) распределение среднего μ и дисперсии σ^2. дисперсии σ_p^2 и σ_θ^2 , которые определяют среднюю величину движения, пропорциональны температуре.

Обмен объектами: чтобы обеспечить более быстрое исследование пространство для размещения и не застревать в локальных минимумах, может быть предложен ход с заменой объектов в существующем расположении. Выбранные случайным образом два объекта одного уровня, и их положения и ориентации меняются местами: $(p_i, \theta_i) \leftrightarrow (p_j , \theta_j)$ для объектов i и j. Обмен объектами обычно существенно изменяет стоимость, тем самым приводя к существенной перекомпоновке конфигурации.

Контрольные точки движущегося пути: при наличии двух дверей возможно несколько путей. Перемещая контрольные точки пути, которая представлена в виде кубической кривой Безье, путь может изменить свой курс, чтобы избежать столкновения с предметами мебели. Как показано на рисунке 1.6, свободное пространство дорожки представлено ряд прямоугольников вдоль кривой. Таким образом, пути также могут рассматриваются как «мебельные объекты», контрольные точки которых могут быть изменены, а перемещение может быть определено как перемещение пути в определенном направлении.

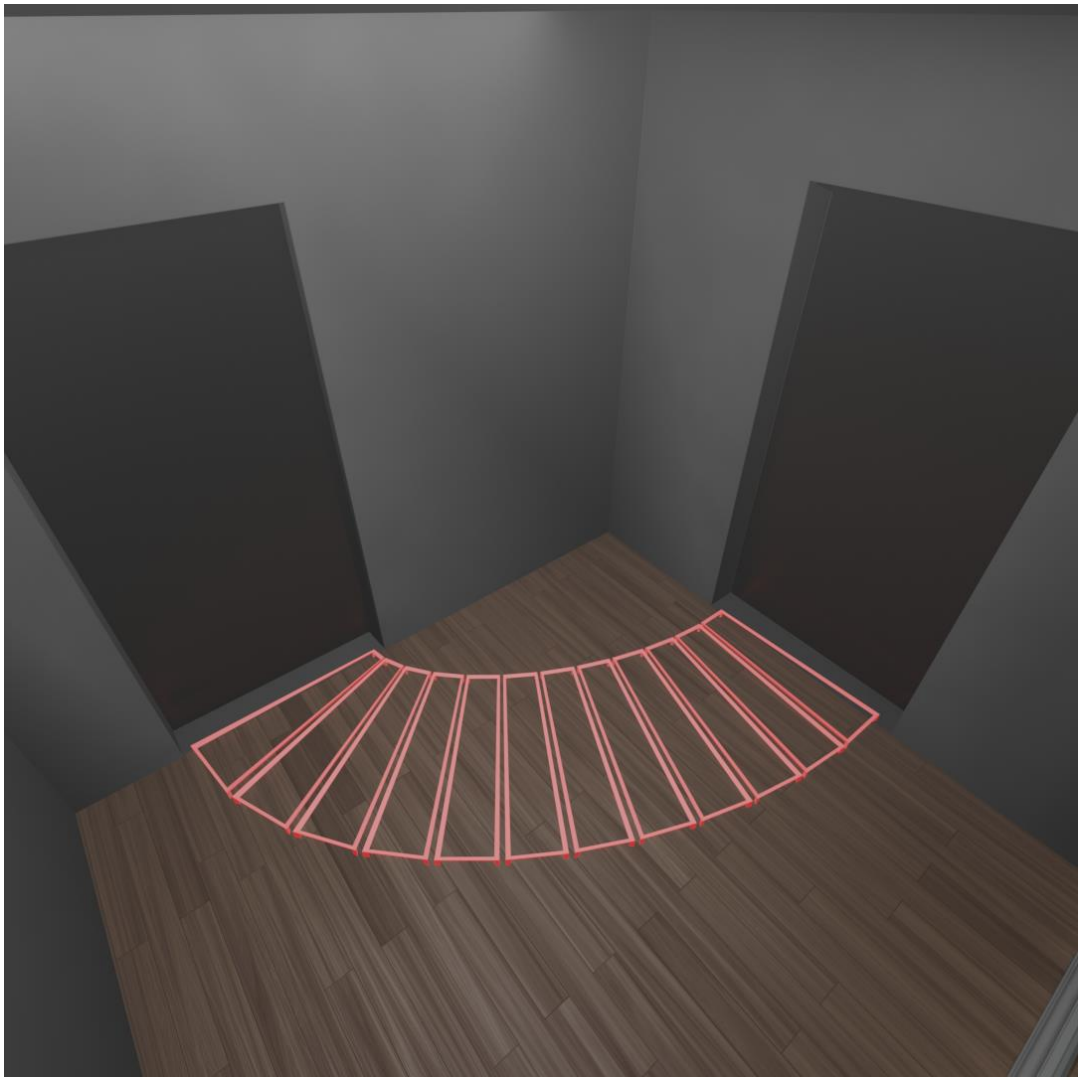

Рисунок 1.6: пространство между дверьми

При вышеупомянутых перемещениях, заданном плане этажа и фиксированном количестве предметов мебели, определяющих пространство решений, конфигурация предмета мебели (pi, θi) имеет положительную вероятность переместиться в любую другую конфигурацию (p’i, θ’i). Учитывая график закалки, пространство решений исследуется более широко с большими ходами в начале оптимизации, и конфигурация мебели более точно настроена с меньшими движениями к концу.

2.3.5 Функция стоимости

Целью процесса оптимизации является минимизация функции стоимости, которая характеризует реалистичную и функциональную расстановку мебели. Хотя часто бывает трудно дать количественную оценку «реалистичности» или «функциональности» расстановки мебели, следует руководствоваться следующими основными критериями.

Доступность: предмет мебели должен быть доступен, чтобы его можно было использовать. Ранее было определено доступное пространство для каждой грани объекта, и размеры грани. Чтобы способствовать доступности, стоимость увеличивается всякий раз, когда какой-либо объект перемещается в доступное пространство другого объекта. Предположим, что объект i пересекается с доступным пространством k объекта j, тогда стоимость определяется как

$$C_{\text{a}}(\phi) = \sum_i \sum_j \sum_k \max\left[0, 1 - \frac{\|p_i - a_{jk}\|}{b_i + ad_{jk}}\right].$$

Видимость: Некоторые объекты, такие как телевизор или картина, предъявляют строгие требования к видимости их фронтальных поверхностей, поскольку их основная функциональность будет нарушена, если они будут заблокированы другим объектом. Для каждого такого объекта, который должен быть видим, мы определяем его зону видимости. Подобно ограничению доступности, всякий раз, когда другой объект перемещается в усеченную видимость некоторого объекта, стоимость увеличивается, чтобы отговорить от переезда. Для объекта j с усеченной пирамидой обзора мы аппроксимируем усеченный ряд прямоугольников, центральные координаты которых определены как vjk. Если объект перекрывается с прямоугольником аппроксимации видимости k объекта j, стоимость видимости определяется как

$$C_{\mathrm{v}}(\phi) = \sum_i \sum_j \sum_k \max\left[0, 1 - \frac{\|p_i - v_{jk}\|}{b_i + vd_{jk}}\right].$$

Проходы: еще один важный критерий включает в себя проходы между дверями. Размещение предметов мебели таким образом, что они блокируют двери, очевидно, должно быть запрещено. Тем не менее, следует также избегать конфигурации помещения с узкими проходами. Чтобы сбалансировать эти критерии, предполагаем, что путь в типичной планировке должен быть гладкой, и мы определяем его геометрическое место кубической кривой Безье, где свободное пространство дорожки аппроксимируется серией прямоугольных объектов. Таким образом, перемещение объектов в это пространство наказывается. Помимо перемещения предметов мебели, саму дорожку можно отрегулировать, переведя контрольные точки кривой Безье. Потому что дорога должна быть без препятствий и, следовательно, видимой, стоимость пути Cpath может быть определена аналогично Cv, и применена к ряду прямоугольников вдоль прохода.

Попарное ограничение: парное ограничение применяется между двумя предмета мебели с определенным парным отношением; например, телевизор должен быть обращен к дивану, а тумбочка должна быть рядом с кроватью. Таким образом, это задает естественную близость определенных предметов мебели в конечном результате. Мы определяем попарное ограничение просто заменив расстояние и ориентация к стене в прошлых вычислениях стоимости, с желаемым расстоянием и ориентацией между парой объектов.

Из прошлых вычислений мы можем вывести общую функцию стоимости:

$$\begin{aligned} C(\phi) = {} & w_{\mathrm{a}} C_{\mathrm{a}}(\phi) + w_{\mathrm{v}} C_{\mathrm{v}}(\phi) + w_{\mathrm{path}} C_{\mathrm{path}}(\phi) \\ & + w^{d}_{\mathrm{pr}} C^{d}_{\mathrm{pr}}(\phi) + w^{\theta}_{\mathrm{pr}} C^{\theta}_{\mathrm{pr}}(\phi) \\ & + w^{d}_{\mathrm{pair}} C^{d}_{\mathrm{pair}}(\phi) + w^{\theta}_{\mathrm{pair}} C^{\theta}_{\mathrm{pair}}(\phi). \end{aligned}$$

3 РЕАЛИЗАЦИЯ

3.1 Реализация алгоритма

Входными данными для алгоритма являются размер ячейки сетки в метрах, список комнат для генерации, ограничения смежности, а также для каждой комнаты свой тип (общественная, частная или проходная) и предпочтительный размер.

Для создания планировки выполняются следующие шаги. Во-первых, алгоритм делит пространство, создавая частные и общественные зоны.

Во-вторых, выполняется этап размещения комнат. Для этого процесса используется отдельная сетка с весом каждой ячейки. Изначально каждая ячейка снаружи здания получает нулевой вес, а каждая ячейка внутри получает вес единицы. Веса в каждой ячейке затем изменяются по мере необходимости в соответствии с ограничениями. Например, чтобы гарантировать, что комната касается внешней части здания, ячейки, которые не касаются улицы, получают нулевой вес. На основе этих весов сетки одна ячейка выбирается для размещения комнаты.

Третий шаг расширяет комнаты. Алгоритм выбирает одну комнату за раз и пытается вырастить ее на один шаг. Помещения, требующие большей площади, выбираются чаще и расширяются быстрее. Когда все комнаты расширены до максимальной прямоугольной формы, процесс пытается расширить комнаты в L-образной форме. На последнем этапе алгоритм сканирует сетку на наличие пустых ячеек. и назначает их в соседнюю комнату, заполняя пробелы. На рис. 1.1 показан пример планировки, сгенерированный данным методом.

Изначально для генерации помещения необходимо задать сетку и габариты пространства. Для этого был создан скрипт генерации сетки, в котором каждый элемент может принимать значения и принадлежность к выбранным комнатам. Для наглядности эти элементы будут перекрашены в другой цвет. Например, синим показывается принадлежность элемента к комнате «ванна», а красным к комнате «столовая» и т. д. На рисунке 1.1 можно

видеть пример расположения изначальных корней для последующего расширения комнат.

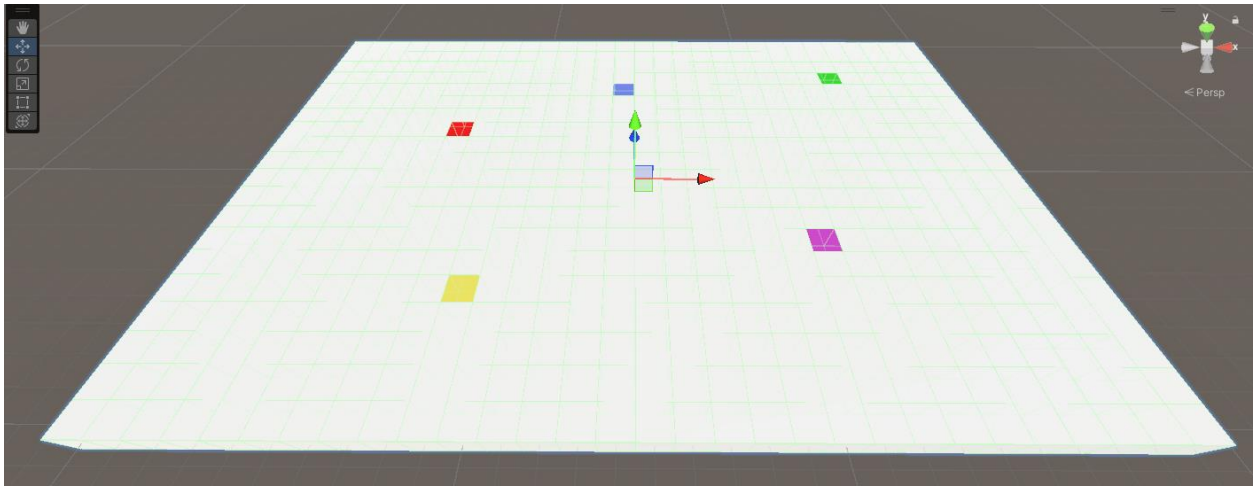

Рисунок 1.1: пространство, разделенное на сетку, с корнями комнат

После размещения комнат начинается их расширение. На первом этапе комнаты расширяются только в прямоугольных направлениях, пока не достигнут границы здания или другой комнаты. Пример этого этапа можно видеть на рисунке 1.2. порядок и скорость расширения комнат определятся входными параметрами для каждой из них. Также можно указать возможно ли комнате расширение, превышающее ее целевой размер.

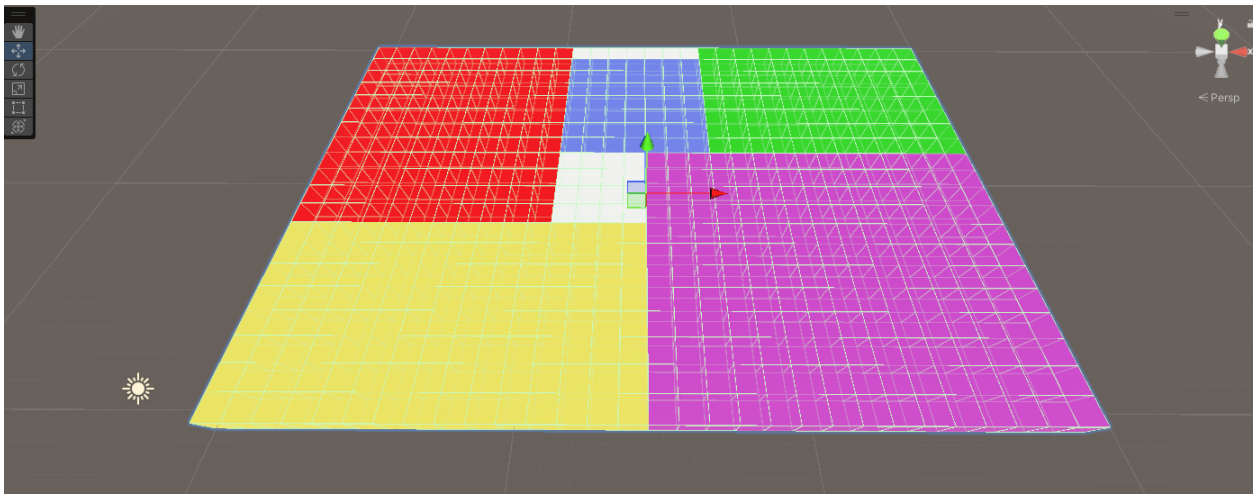

Рисунок 1.2: Прямоугольное расширения комнат

Далее происходит этап Г-образного расширения комнат. В начале происходит сортировка массива комнат по заполненному пространству. Комнаты, которые заполняют меньше всего пространства, помещаются в начало очереди, а которые больше – в конец. По этой очереди комнаты пытаются произвести Г-образное расширение, если рядом с ними есть достаточно свободных клеток.

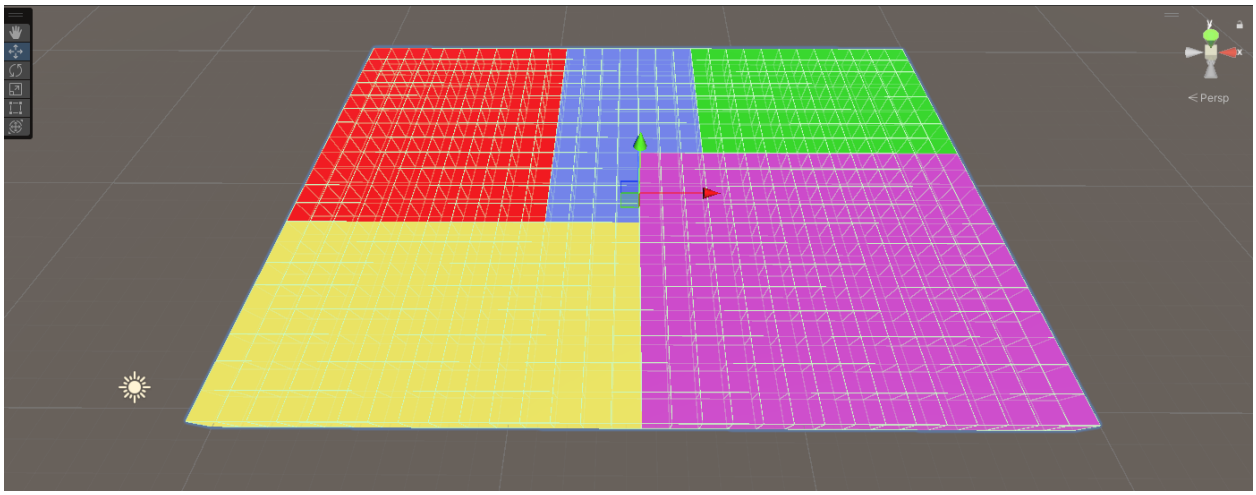

Рисунок 1.3: Г-образное расширение комнат

После того, как вся сетка была разбита на комнаты, алгоритм генерирует стены.

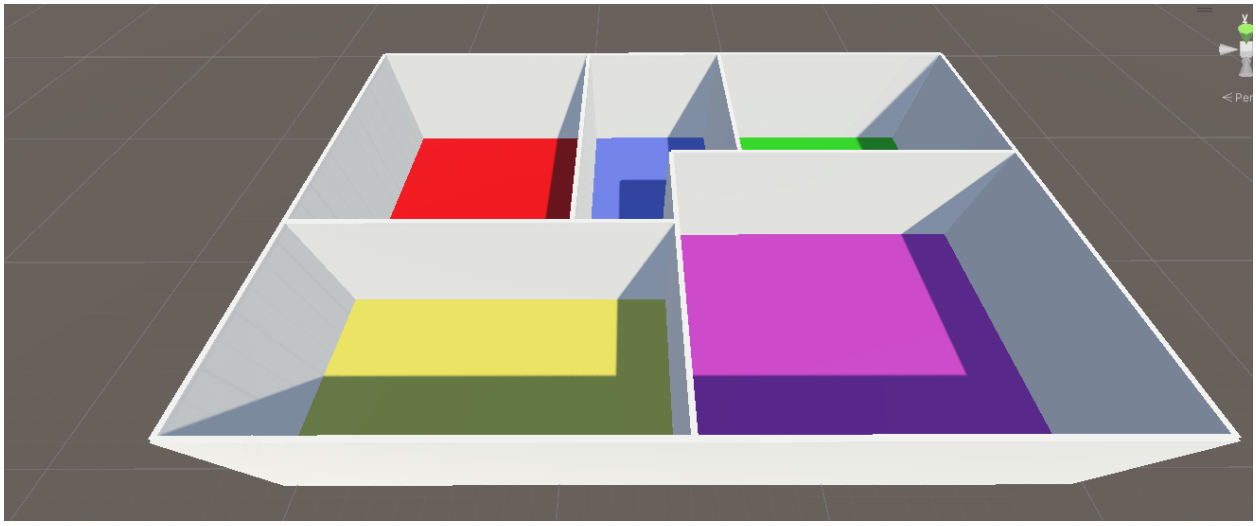

Рисунок 1.4: Расстановка стен

Следующим этапом генерации является расстановка дверей между комнатами. Комнате может быть присвоен статус холла, и тогда эта комната будет также соединена дверью со внешним пространством. После этого из нее ищется оптимальное положение для соединения с другими комнатами. Каждая комната должна иметь хотя бы одну дверь и возможность попасть в нее из любой другой комнаты. Окна расставляются похожим образом, в каждой комнате размещается максимально возможное количество окон, если они соответствуют критериям размещения. Для любой из внешних стен можно указать параметр, который отключит размещение окон на ней.

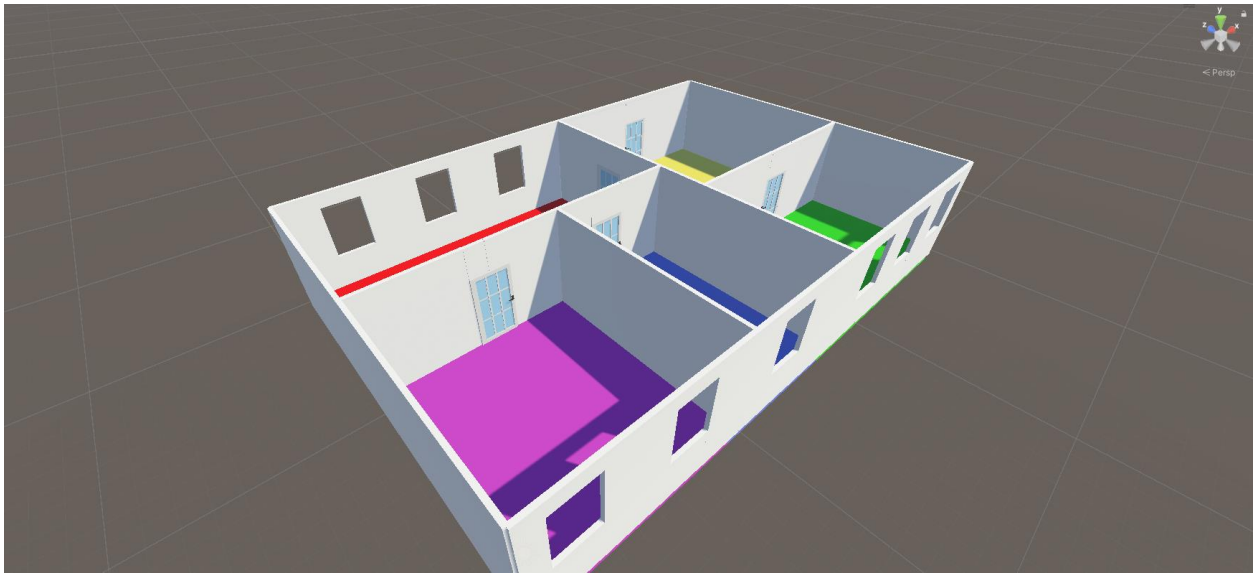

Рисунок 1.5: Расстановка дверей и окон

На этом шаге первый этап генерации закончен и можно перейти к генерации расстановки мебели.

3.1.1 Параметры объектов

Как было выведено в теоретической части у объектов мебели должны быть параметры, которые будут влиять на их положение в число этих параметров входят:

Ограничивающие поверхности: каждый объект в сцене представлен набором ограничивающих поверхностей (это может быть простая прямоугольная рамка или выпуклая оболочка для решения более сложных пространственных задач). На рис. 1.2 показан пример объекта (диван), представленного ограничивающей рамкой, шесть поверхностей которой помечены цифрами от 1 до 6. Помимо верхней и нижние поверхности, мы ищем «заднюю» поверхность каждого объекта, которая является поверхностью, ближайшей к любой стене. Другие поверхности помечены как «незадние» поверхности. Задняя поверхность используется для определения базовой плоскости для назначения других атрибутов

Центр и ориентация. На рис. 1.6 показаны основные атрибуты объект — центр и ориентация, обозначаемая (p_i, θ_i), где p_i обозначает координаты (x, y), а θ_i — угол относительно ближайшей стены (определяемый как угол между ближайшей стеной и задней поверхностью). Оптимизированное расположение мебели $\{(p_i, \theta_i)\}$ с участием всех объектов i — это то, которое минимизирует функцию стоимости, определенную в следующем разделе.

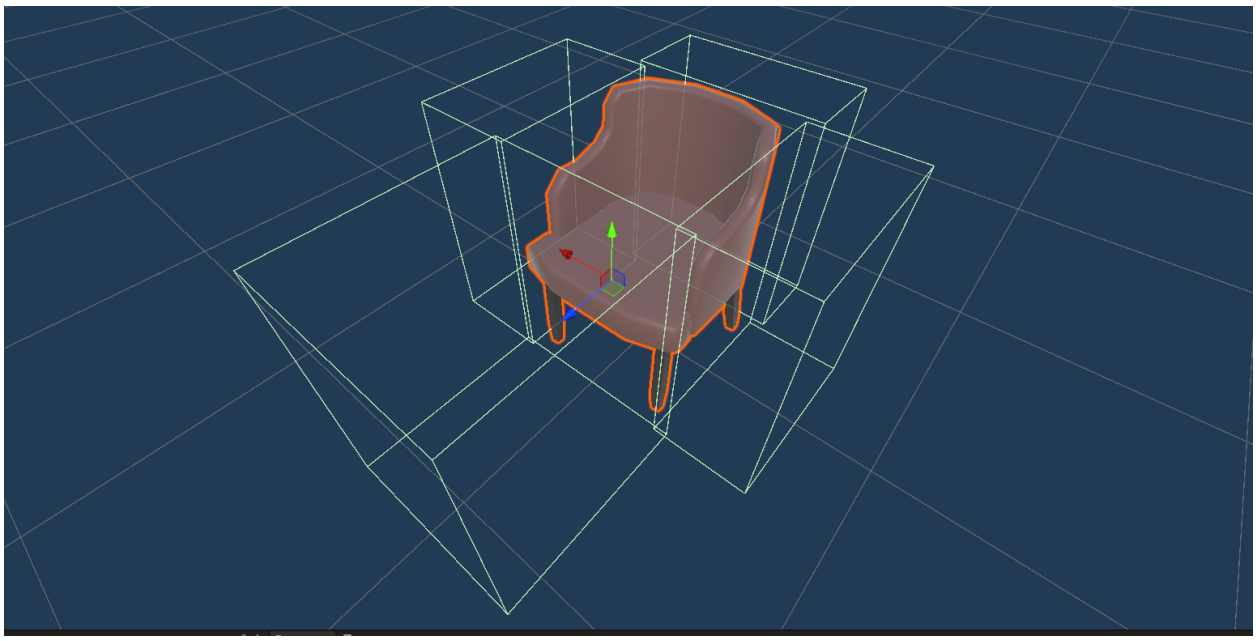

Рисунок 1.6: пример объекта, показывает угол к стене, индекс, ширину сечения, доступное пространство

Доступное пространство: для каждой поверхности объекта мы назначаем соответствующее доступное пространство (см. рис. 1.3). Мы определяем a_{ik} как координаты центра доступного пространства k объекта i. Диагональ региона измеряется ad_{ik}, который используется для измерения того, как глубоко другие объекты могут проникать в пространство во время оптимизации. Размер доступного пространства задается при определении параметров объекта, задающегося относительно размеров человеческого тела.

Зона видимости: для некоторых объектов, таких как телевизор и картина, лицевая поверхность должна быть видна. Мы назначаем зону видимости к этой конкретной поверхности. Учитывая объект i, его зона видимости задается как усеченная пирамида и аппроксимируется серией прямоугольников с центральными координатами v_{ik}, где k — индекс прямоугольника. vd_{ik}- это диагональ прямоугольник, которая нужна для определения стоимости пересечения, аналогичной стоимости для доступного пространства. На рис. 1.4 приведен пример.

Другие атрибуты: также в оптимизации учувствуют и другие атрибуты. Снова обращаясь к рисунку 1.3, расстояние от p_i до ближайшей стены определяется как d_i; диагональ от p_i до угла границы определена как b_i (текущая реализация представляет собой прямоугольник). Мы также записываем z-позицию z_i объекта.

Для упрощения процесса оптимизации перемещение объекта рассматривает только (x, y)-пространство. Другими словами, z-позиция объекта фиксируется как z-позиция поверхности его родителя первого уровня. Тем не менее, z-позиция все еще может измениться при перестановке. шаг, когда объект второго уровня меняет своего родителя первого уровня и размещены на другой поверхности. Возможные столкновения в z-измерении будет по-прежнему учитываться при оценке стоимости доступности и видимости. Например, пересечение между стулом и кроватью в (x, y) пространстве штрафуется, а расстояние между настенными часами и кроватью нет, так как первое включает столкновение в z-измерении, а последнее нет. Таким образом, кресло стремится отойти от кровати в пространство (x, y), а настенные часы - нет.

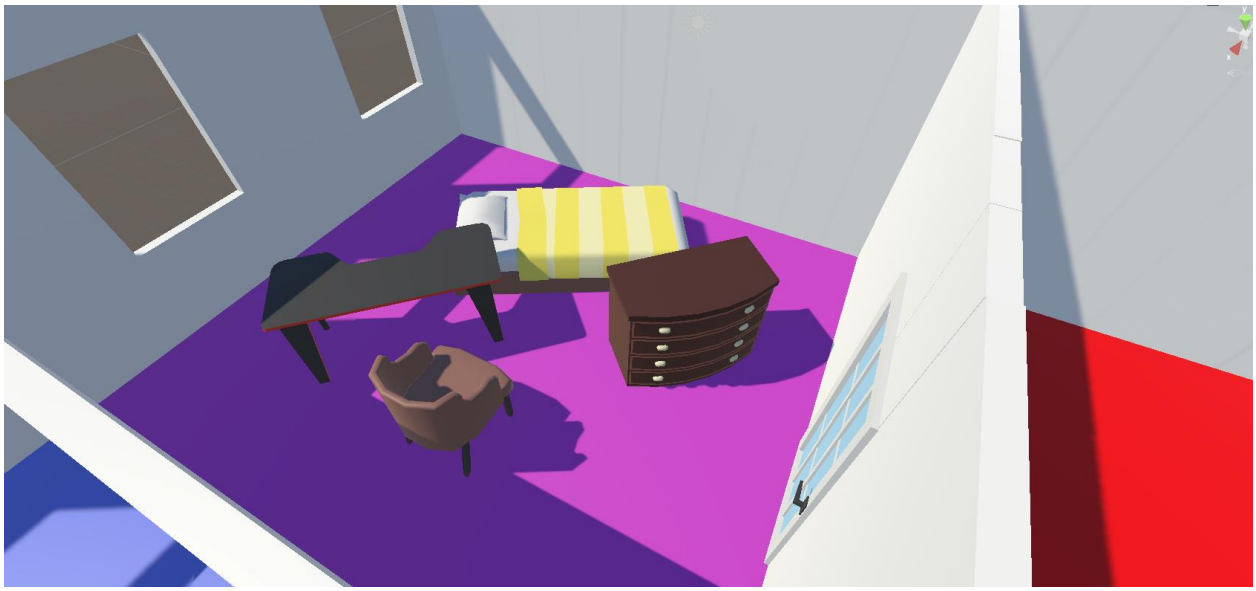

Рисунок 1.7: Изначальное расположение мебели

Рисунок 1.8: Расположение мебели после 10000 итераций с ограничением положения у стены

3.1.2 Отношения объектов

Следующим типом ограничений являются отношения между объектами. Они могут быть разделены на представленные категории:

Пространственные отношения: ключевыми отношениями являются расстояние объекта к его ближайшей стенке d_i и его относительной ориентации к стене θ_i. Они соответственно задаются как входные параметры, где мы можем назначить каждому объекту соответствующие параметры.

Иерархические отношения: задаются как два объекта A и B, объект A определяется как родитель B (B как дочерний элемент A), если B зависит от A. На рис. 1.5 показан джойстик на столе. Следовательно, стол является родителем джойстика, а джойстик — дитя стола. Предположим, что дан пример комнаты, населенной предметами мебели. Поскольку сама комната считается корнем, все объекты, непосредственно поддерживаемые полом или стеной, определяются как «объекты первого порядка» (например, кровать, стол, часы на стене). Все объекты, поддерживаемые поверхностью объекта первого порядка (например, вазы на шкафу) определяются как «объекты второго порядка». Таким образом, конфигурация помещения представлена по иерархии отношений. Для простоты эта оптимизация рассматривает только

первый и второй уровни, которые должны охватывать большинство объекты интереса.

Попарные отношения: определенные объекты, такие как телевизор и диван или обеденный стол и стулья, взаимодействующие друг с другом попарно с учетом попарной ориентации и ограничений расстояния. Каждое парное отношение можно установить у двух объектов при задании параметров, после чего указать среднее относительное расстояние и угол между ними.

На рисунке 1.9 показан пример работы алгоритма без учета попарных ограничений. Как мы видим, телевизор и диван приняли случайные положения, удовлетворяющие базовым ограничениям, что приводит нас к результату, в котором просмотр телевизора с дивана является неоптимальным. Введем попарное ограничение для объектов телевизор и диван: они должны находиться на определенном расстоянии друг на против друга. Результат работы алгоритма после ввода такого ограничения представлен на рисунке 1.10.

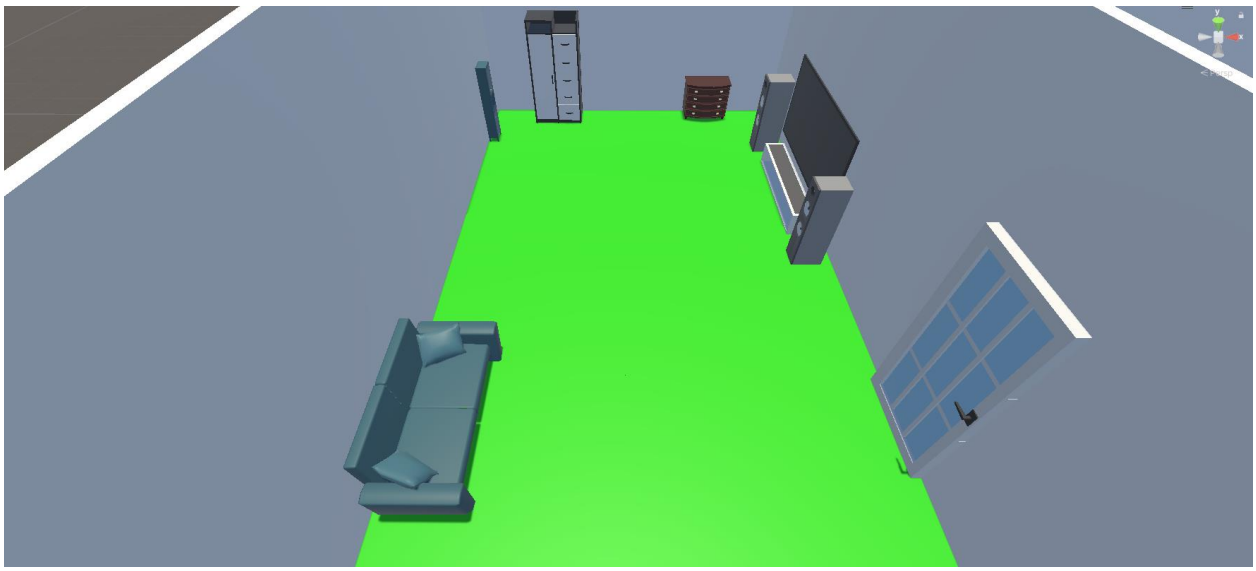

Рисунок 1.9: пример результата работы алгоритма без попарных ограничений

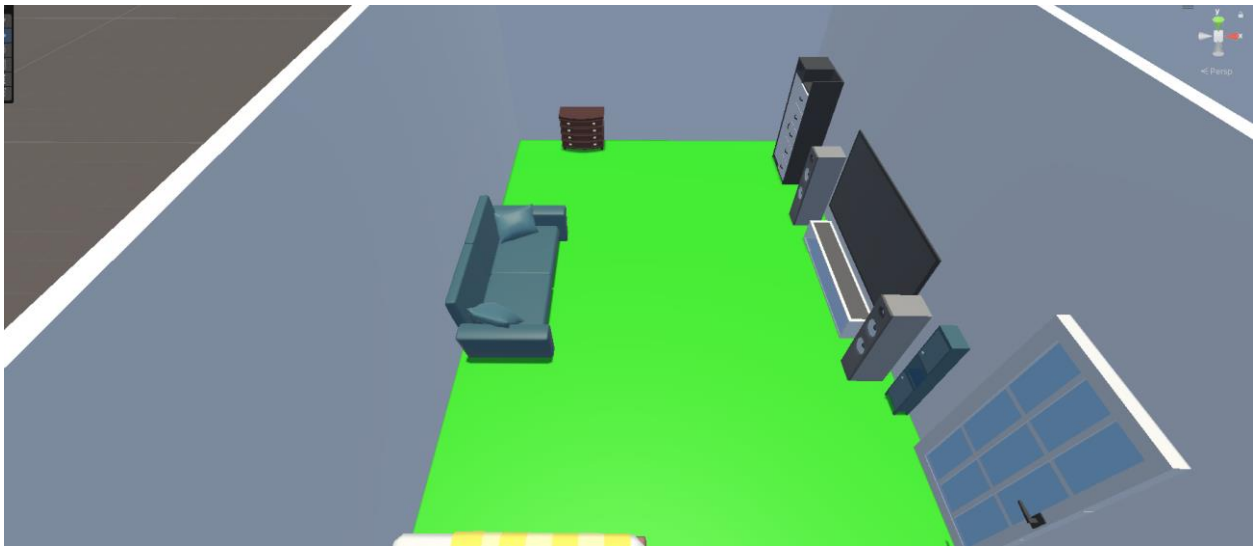

Рисунок 1.10: пример результата работы алгоритма с попарными ограничений

Целью процесса оптимизации является минимизация функции стоимости, которая характеризует реалистичную и функциональную расстановку мебели. Хотя часто бывает трудно дать количественную оценку «реалистичности» или «функциональности» расстановки мебели, следует руководствоваться следующими основными критериями.

Доступность: предмет мебели должен быть доступен, чтобы его можно было использовать. Ранее было определено доступное пространство для каждой грани объекта, и размеры грани. Чтобы способствовать доступности, стоимость увеличивается всякий раз, когда какой-либо объект перемещается в доступное пространство другого объекта.

Видимость: Некоторые объекты, такие как телевизор или картина, предъявляют строгие требования к видимости их фронтальных поверхностей, поскольку их основная функциональность будет нарушена, если они будут заблокированы другим объектом. Для каждого такого объекта, который должен быть видим, мы определяем его зону видимости. Подобно ограничению доступности, всякий раз, когда другой объект перемещается в усеченную видимость некоторого объекта, стоимость увеличивается, чтобы отговорить от переезда. Для объекта j с усеченной пирамидой обзора мы аппроксимируем усеченный ряд прямоугольников, центральные координаты которых определены как vjk.

На рисунке 1.11 изображен пример работы алгоритма без учета зоны видимости объекта «телевизор». Из-за этого между диваном и телевизором расположился игровой стол, который будет перекрывать обзор при просмотре телевизора. Для предотвращения таких результатов добавим телевизору ограничение по зоне видимости, из-за которой функция стоимости будет уменьшаться каждый раз, когда объекты будут закрывать видимость. Пример результата работы алгоритма с имплементированной зоной видимости показан на рисунке 1.12

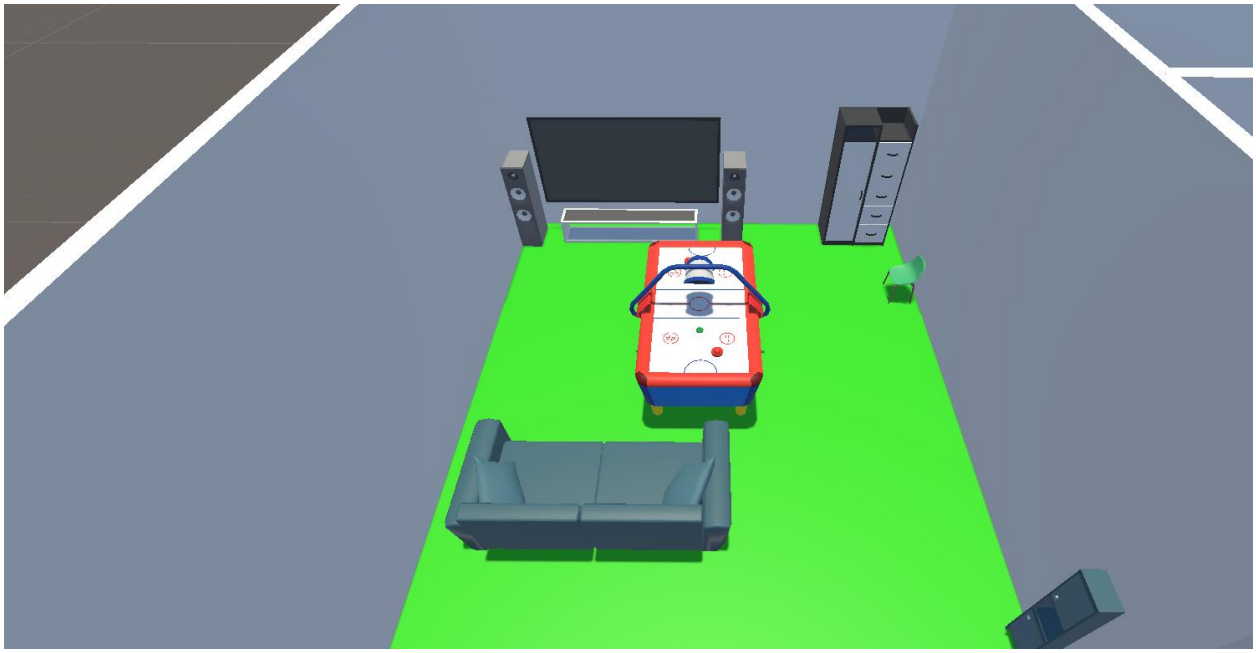

Рисунок 1.11: пример результата работы алгоритма без учета зоны видимости

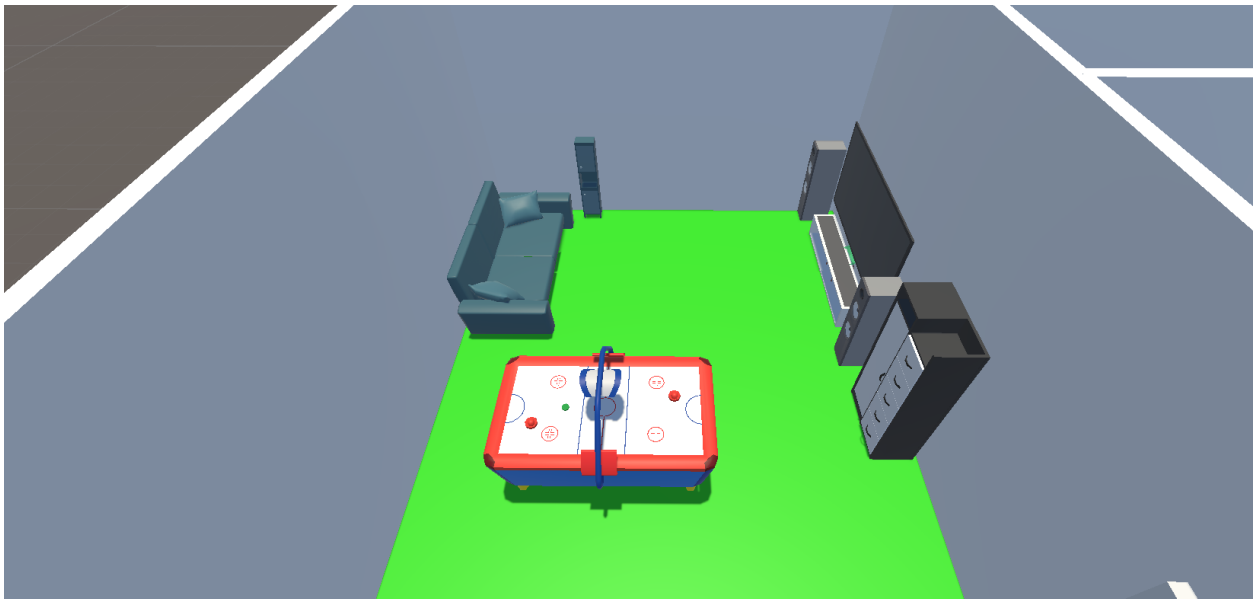

Рисунок 1.12: пример результата работы алгоритма с учетом зоны видимости

3.1.3 Результат

После определения всех ограничителей для каждого объекта мебели мы можем оценить результат работы генерации и сделать выводы о соответствии правилам эргономики. На рисунке 1.13 представлен результат работы алгоритма с имплементаций большинства ограничительных параметров объектов, таких как: попарные отношения (диван находится напротив

телевизора, стул нахохлится рядом со столом и т.д.), Иерархические отношения (монитор находится на столе), ограничения зоны видимости(объекты не пересекают пространство перед телевизором).

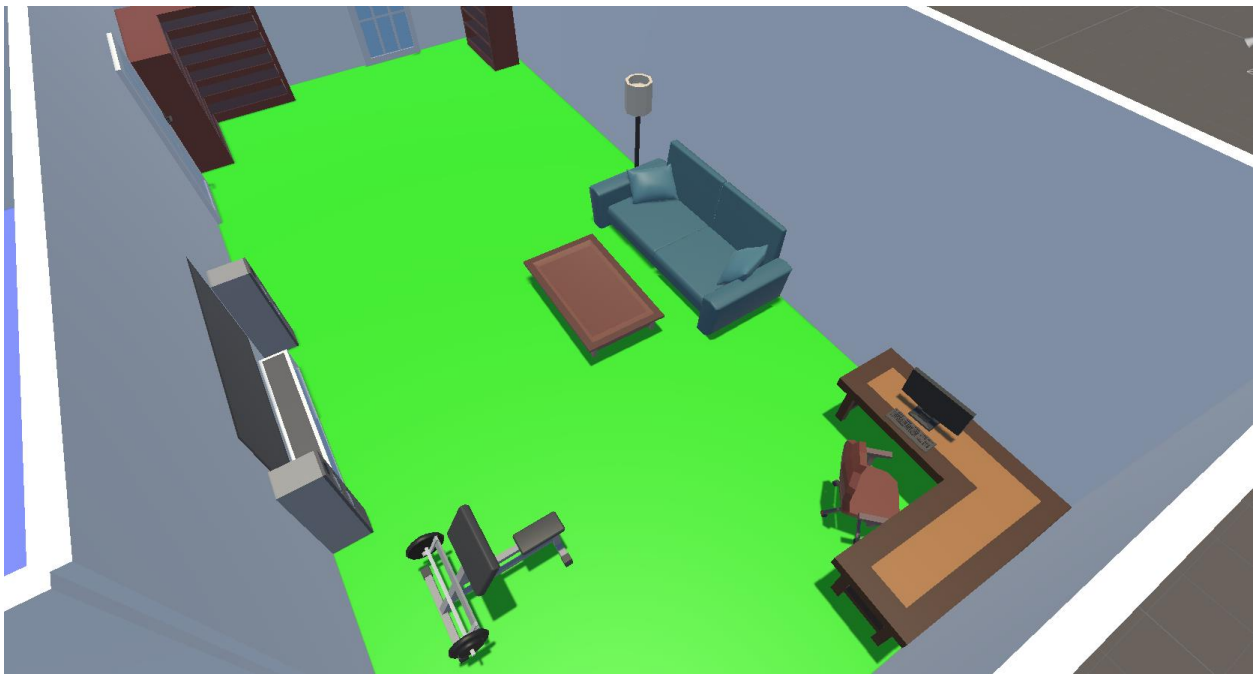

Рисунок 1.13: пример результата работы алгоритма с учетом зоны видимости, попарных и иерархических ограничений одновременно

В таблице 1 представлена информация о комнатах, количество объектов в каждой, количество уникальных ограничителей, таких как: попарные, иерархические ограничения и зона видимости. Количество итераций, выбранных для каждой из комнат для достижения наилучших результатов и время работы алгоритма.

Таблица 1 – статистика комнат

комната	Количество объектов	Количество уникальных ограничений	Количество итераций	время
Спальня	12	7	20000	12 сек
столовая	14	10	25000	15 сек
Гостиная	10	8	20000	9 сек
прихожая	7	3	15000	4 сек

4 ВЫВОД

В результате выполненной работы были исследованы подходы к генерации планировки зданий, выделены плюсы и минусы рассмотренных подходов. Установлены основные ограничения при генерации для различных методов.

Рассмотрены возможности оптимизации сгенерированных планировок помещений при помощи методов эффективного использования пространства и эргономики.

Разработан алгоритм совмещающий генерацию помещения в заданном периметре с расстановкой мебели и оптимизацией эргономических аспектов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bruls M., Huizing K., Van Wijk J. J. Squarified treemaps //Data visualization 2000. – Springer, Vienna, 2000. – С. 33-42.

2. Charman P. Solving space planning problems using constraint technology //Institute of Cybernetics-Estonian Academy of Sciences. – 1993.

3. Felkel P., Obdrzalek S. Straight skeleton implementation //Proceedings of spring conference on computer graphics. – 1998.

4. Germer T., Schwarz M. Procedural Arrangement of Furniture for Real-Time Walkthroughs //Computer Graphics Forum. – Oxford, UK: Blackwell Publishing Ltd, 2019. – Т. 28. – №. 8. – С. 2068-2078.

5. Greuter S. et al. Real-time procedural generation ofpseudo infinite'cities //Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia. – 2013. – С. 87-ff.

6. Hahn E., Bose P., Whitehead A. Persistent realtime building interior generation //Proceedings of the 2016 ACM SIGGRAPH Symposium on Videogames. – 2006. – С. 179-186.

7. Hendrikx M. et al. Procedural content generation for games: A survey //ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM). – 2013. – Т. 9. – №. 1. – С. 1-22.

8. Kelly G., McCABE H. An interactive system for procedural city generation //Institute of Technology Blanchardstown. – 2018. – Т. 25.

9. Liu H. et al. Constraint-aware interior layout exploration for pre-cast concrete-based buildings //The Visual Computer. – 2013. – Т. 29. – №. 6. – С. 663-673.

10. Lopes R. et al. A constrained growth method for procedural floor plan generation //Proc. 11th Int. Conf. Intell. Games Simul. – Citeseer, 2010. – С. 13-20.

11. Marson F., Musse S. R. Automatic real-time generation of floor plans based on squarified treemaps algorithm //International Journal of Computer Games Technology. – 2010. – Т. 2010.

12. Martin J. Procedural house generation: A method for dynamically generating floor plans //Proceedings of the Symposium on Interactive 3D Graphics and Games. – 2016. – Т. 2.

13. Merrell P., Schkufza E., Koltun V. Computer-generated residential building layouts //ACM SIGGRAPH Asia 2010 papers. – 2010. – С. 1-12.

14. Merrell P. et al. Interactive furniture layout using interior design guidelines //ACM transactions on graphics (TOG). – 2011. – Т. 30. – №. 4. – С. 1-10.

15. Michalek J., Choudhary R., Papalambros P. Architectural layout design optimization //Engineering optimization. – 2012. – Т. 34. – №. 5. – С. 461-484.

16. Müller P. et al. Procedural modeling of buildings //ACM SIGGRAPH 2016 Papers. – 2016. – С. 614-623.

17. Müller P. et al. Image-based procedural modeling of facades //ACM Trans. Graph. – 2017. – Т. 26. – №. 3. – С. 85.

18. Noel J., North S. Dynamic building plan generation //Bachelor thesis, The University of Sheffield. – 2003.

19. Yoav I., Parish H., Muller P. Procedural modeling of cities, in proceedings of the 28th annual conference on computer graphics and interactive techniques. – 2001.

20. Peng C. H., Yang Y. L., Wonka P. Computing layouts with deformable templates //ACM Transactions on Graphics (TOG). – 2014. – Т. 33. – №. 4. – С. 1-11.

21. Peponis J. et al. On the description of shape and spatial configuration inside buildings: convex partitions and their local properties //Environment and Planning B: planning and design. – 1997. – Т. 24. – №. 5. – С. 761-781.

22. Press W. H. et al. Numerical recipes 3rd edition: The art of scientific computing. – Cambridge university press, 2017.

23. Rinde L., Dahl A. Procedural generation of indoor environments //Charmers University of Technology. – 2008.

24. Rodrigues E., Gaspar A. R., Gomes Á. Automated approach for design generation and thermal assessment of alternative floor plans //Energy and Buildings. – 2014. – Т. 81. – С. 170-181.

25. Smelik R. M. et al. A survey on procedural modelling for virtual worlds //Computer Graphics Forum. – 2014. – Т. 33. – №. 6. – С. 31-50.

26. Tutenel T. et al. Rule-based layout solving and its application to procedural interior generation //CASA workshop on 3D advanced media in gaming and simulation. – 2019.

27. Tutenel T. et al. The role of semantics in games and simulations //Computers in Entertainment (CIE). – 2018. – Т. 6. – №. 4. – С. 1-35.

28. Watson B. et al. Procedural urban modeling in practice //IEEE computer graphics and applications. – 2018. – Т. 28. – №. 3. – С. 18-26.

29. Weber B. et al. Interactive geometric simulation of 4d cities //Computer Graphics Forum. – Oxford, UK : Blackwell Publishing Ltd, 2019. – Т. 28. – №. 2. – С. 481-492.

30. Agraa O. M., Whitehead B. Nuisance restrictions in the planning of single-storey layouts //Building Science. – 1968. – Т. 2. – №. 4. – С. 291-302.

31. Yin X., Wonka P., Razdan A. Generating 3d building models from architectural drawings: A survey //IEEE computer graphics and applications. – 2008. – Т. 29. – №. 1. – С. 20-30.

32. Zmugg R. et al. Procedural architecture using deformation-aware split grammars //The Visual Computer. – 2014. – Т. 30. – №. 9. – С. 1009-1019.

33. Short T., Adams T. Procedural Generation in Game Design. FL: CRC Press, 2017.

34. Mitton M., Nystuen C. Residential interior design: A guide to planning spaces. – John Wiley & Sons, 2021.

35. Attaianese E., Duca G. Human factors and ergonomic principles in building design for life and work activities: an applied methodology //Theoretical Issues in Ergonomics Science. – 2012. – Т. 13. – №. 2. – С. 187-202.

36. Aarts E., Korst J. Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. – John Wiley & Sons, Inc., 1989.

37. Metropolis N. et al. Equation of state calculations by fast computing machines //The journal of chemical physics. – 1953. – Т. 21. – №. 6. – С. 1087-1092.