Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б07-мм

# Реализация процедурной генерации интерьера комнат в 3D-игре на Unity

***Савельева Полина Андреевна***

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
доцент кафедры системного программирования, к.т.н. Ю. В. Литвинов

Консультант:
студент 3-го курса СПбГУ, Р. А. Береснёв

Санкт-Петербург
2023

# Оглавление

# 1. Введение

Местом действия игры Time Reactor[1] является бесконечная лестница, на этажах которой случайным образом располагаются помещения одного из трёх типов: лаборатория, компьютерный центр и склад технического оборудования. Поскольку число комнат, доступных для изучения, не ограничено, то ручное заполнение помещений может вызвать ряд трудностей при создании сотен и тысяч различных интерьерных решений. Кроме того, чем дальше игрок будет продвигаться по лестнице, тем больше уровни будут казаться ему однотипными и похожими друг на друга. Создание большого количества разнообразных интерьеров возможно с помощью методов *процедурной генерации*.

Под процедурной генерацией контента (Procedural Content Generation, *PCG*) понимается наполнение игрового мира с помощью алгоритмов при прямом или косвенном участии пользователя [2]. Методы PCG широко используются при проектировании и разработке игр [1] — от генерации целых уровней в Spelunky (Mossmouth 2009) и планет в No Man's Sky (Hello Games 2016) до характеров персонажей и сюжетов в RimWorld (Ludeon Studios 2013).

В книге «Procedural generation in game design» Short Tanya и Adams Tarn приводят ряд причин, побуждающих разработчиков использовать процедурную генерацию в своих проектах. Исходя из них, можно выделить основные мотивы для применения методов PCG в игре Time Reactor для генерации интерьера комнат:

- создание разнообразного контента в масштабах, недостижимых вручную;
- отсутствует необходимость в хранении сгенерированных данных;
- использование PCG может занять сравнительно меньше времени, чем ручное создание того же объёма данных;
- повторное использование генераторов.

[1] https://github.com/RuslanBeresnev/Time-Reactor-Game (дата обращения: 27 сентября 2023 г.).

В свою очередь, *процедурная генерация помещений* направлена на создание виртуальной среды, приближенной к реальной, с помощью алгоритмов PCG. Процедурно генерируемое помещение строится из описания объектов, характерных для данного пространства, и связей между ними. К последнему можно отнести иерархические паттерны (посуда на кухне располагается строго на полках, стулья в кабинете за письменным столом) и функциональные требования (экран телевизора находится в зоне видимости и не заставлен другими предметами).

Для формирования реалистичного игрового мира, система генерации интерьеров должна отвечать и другим требованиям. В частности, активировать алгоритм процедурной генерации следует в наиболее подходящий момент, например, при попадании комнаты в зону видимости игрока (в случае Time Reactor — во время инициализации всей комнаты). Сам процесс генерации обязан происходить достаточно быстро, чтобы избежать ситуаций, когда игрок заходит в комнату ещё до окончания расстановки мебели. Кроме того, интерьер должен оставаться неизменным при каждом следующем посещении.

Таким образом, необходимо не только реализовать алгоритм для процедурной генерации реалистичного интерьера комнат, но и обеспечить его эффективную интеграцию с остальными игровыми процессами, а также неизменность результата при многократном вызове.

# 2. Постановка задачи

Целью данной работы является реализация процедурной генерации интерьера комнат в игре Time Reactor на Unity. Для её выполнения были поставлены следующие задачи.

- Реализация .NET библиотеки для процедурной генерации интерьера комнат.
- Реализация процедурной генерации интерьера комнат в игре Time Reactor на Unity с использованием разработанной библиотеки.
- Проведение апробации среди заинтересованных игроков.

# 3. Обзор (обязателен к новому году)

*Обзор должен быть.* Здесь нужно писать, что индустрия и наука уже сделали по вашей теме. Он нужен, чтобы Вы случайно не изобрели какой-нибудь велосипед.

По-английски называется related works или previous works.

Если Ваша работа является развитием предыдущей и плохо понимаема без неё, то обзор должен идти в начале. Если же Вы решаете некоторую задачу новым интересным способом, то если поставить обзор в начале, то читатель может устать, пока доберется до вашего решения. В этом случае уместней поставить обзор после описания Вашего подхода к проблеме.

В обзоре необходимо ссылаться на работы других людей. В данном шаблоне задумано, что литература будет указываться в файле `vkr.bib`. В нём указываются пункты литературы в формате BibTeX, а затем на них можно ссылаться с помощью `\cite{...}`. Та литература, на которую Вы сошлетесь, попадет в список литературы в конце документа. Если не сошлетесь — не попадёт. Спецификацию в формате BibTeX почти никогда (для второго курса — никогда), не нужно придумывать руками. Правильно: находить в интернете описание цитируемой статьи[2], копировать цитату с помощью кнопки “Export Citation” и вставлять в BibTeX файл. Если так не делать, но оформление литературы будет обрастать багами. Например, BibTeX по особенному обрабатывает точки, запятые и `and` в списке авторов, что позволяет ему самому понимать, сколько авторов у статьи, и что там фамилия, что — имя, а что — отчество.

В обзоре и в остальном тексте вы наверняка будете использовать названия продуктов или языков программирования. Для них рекомендуется (в файле `preamble2.tex`) задать специальные команды, чтобы писать сложные названия правильно и одинаково по всему документу. Написать с ошибкой название любимого языка программирования научного руководителя — идеальный вариант его выбесить.

[2]Например, https://dl.acm.org/doi/10.1145/3408995 (дата доступа: 17 декабря 2022 г.).

# 4. Background (опционально)

Здесь пишется некоторая дополнительная информация о том, зачем делается то, что делается.

Например, в работе придумывается какой-то новый метод решения формул в SMT в теориях с числами. Без каких-то дополнительных пояснений будет казаться, что работа состоит из жестокого «матана» и совсем не по теме кафедры системного программирования. Поэтому, в данном разделе стоит рассказать, что все эти методы примеряются для верификации в проекте V♯, и поэтому непосредственно связаны с тематикой кафедры.

# 5. Метод

Реализация в широком смысле: что таки было сделано. Скорее всего самый большой раздел.

*Крайне желательно* к Новому году иметь что-то, что сюда можно написать.

Для понимания того как курсовая записка (отзыв по учебной практике/ВКР) должна писаться, можно посмотреть видео ниже. Они про научные доклады и написание научных статей, учебные практики и ВКР отличаются тем, что тут есть требования отдельных глав (слайдов) со списком задач и списком результатов. Но даже если вы забьёте не требования специфичные для ВКР, и соблюдете все рекомендации ниже, получившиеся скорее всего будет лучше чем первая попытка 99% ваших однокурсников.

1. «Как сделать великолепный научный доклад» от Саймона Пейтона Джонса [**?**] (на английском).

2. «Как написать великолепную научную статью» от Саймона Пейтона Джонса [**?**] (на английском).

3. «Как писать статьи так, чтобы люди их смогли прочитать» от Дэрэка Драйера [**?**] (на английском). По предыдующей ссылке разбираются, что должно быть в статье, т.е. как она должна выглядеть на высоком уровне. Тут более низкоуровнево изложено как должны быть устроены параграфы и т.п.

4. Ещё можно посмотреть How to Design Talks [**?**] от Ranjit Jhala, но я думаю, что первых трех ссылок всем хватит.

## 5.1. Некоторые типичные ошибки

Здесь мы будем собирать основные ошибки, которые случаются при написании текстов. В интернетах тоже можно найти коллекции типичных косяков[3].

[3] https://www.read.seas.harvard.edu/~kohler/latex.html (дата доступа: 16 декабря 2022 г.).

Рекомендуется по-умолчанию использовать красивые греческие буквы $\sigma$ и $\varphi$, а именно $\varphi$ вместо $\phi$. В данном шаблоне команды для этих букв переставлены местами по сравнению с ванильным TEX'ом.

Также, если работа пишется на русском языке, необходимо, чтобы работа была написана на *грамотном* русском языке даже, если автор, из ближнего зарубежья[4]. Это включает в себя:

- разделы должны оформляться с помощью `\section{...}`, а также `\subsection` и т. п.; не нужно пытаться нумеровать вручную;
- точки после окончания предложений должны присутствовать;
- пробелы после запятых и точек, в конце слова перед скобкой;
- неразрывные пробелы, там, где нужны пробелы, но переносить на другую строку нельзя, например `т.~е.` или `А.~Н.~Терехов`;
- дефис, там где нужен дефис (обозначается с помощью одиночного «минуса» (англ. dash) на клавиатуре);
- двойной дефис, там где он нужен; а именно при указании промежутка в цифрах: в 1900–1910 г. г., стр. 150–154;
- тире (т. е. `---` — тройной минус) на месте тире, а не что-то другое; в русском языке тире не может «съезжать» на новую строку, поэтому стоит использовать такой синтаксис: `До~--- после`;
- правильные двойные кавычки должны набираться с помощью пакета `csquotes`: для основного языка в `polyglossia` стоит использовать команду `\enquote{текст}`, для второго языка стоит использовать `\foreignquote{язык}{текст}`;
- все перечисления должны оформляться с помощью `\enumerate` или `\itemize`; пункты перечислений должны либо начинаться с заглавной буквой и заканчиваться точкой, либо начинаться со

[4]Теоретически, возможен вариант написания текстов на английском языке, но это необходимо обсудить в первую очередь с научным руководителем.

строчной и закачиваться точкой с запятой; последний пункт перечислений всегда заканчивается точкой.

- Перед выкладкой финальной версии необходимо просмотреть лог LaTeX'а, и обратить внимание на сообщения вида *Overfull \hbox (1.29312pt too wide) in paragraph*. Обычно, это означает, что текст выползает за поля, и надо подсказать, как правильно слова на слоги, чтобы перенос произошел автоматически. Это делается, например, так: `соломо\-волокуша`.

## 5.2. Листинги, картинка и прочий «не текст»

Различный «не текст» имеет свойство отображаться не там, где он написан в текстовом в LaTeX, поэтому у него должна быть самодостаточная понятная подпись `\caption{...}`, уникальная метка `\label{...}`, чтобы на неё можно было бы ссылаться в тексте с помощью `\ref{...}`. Ниже вы сможете увидеть таблицу 1, на которую мы сослались буквально только что.

«Не текста» может быть довольно много, чтобы не засорять корневую папку, хорошим решением будет складывать весь «не текст» в папку `figures`. Заклинание `\includegraphics{}` уже знает этот путь и будет искать там файлы без указания папки. Команда `\input{}` умеет ходит по путям, например `\input{figures/my_awesome_table.tex}`. Кроме того, листинги кода можно подтягивать из файла с помощью команды `\lstinputlisting{file}`.

**Листинг 1: Название для листинга кода. Достаточно длинное, чтобы люди, которые смотрят картинку сразу после названия статьи (т. е. все люди), смогли разобраться и понять к чему в статье листинги, картинки и прочий «не текст».**

```
let x = 5 in x+1
```

```
#include <stdio.h>
#include <math.h>

int main(void)
{
  double c = -1;
  double z = 0;

  printf ("For␣c␣=␣%lf:\n", c);
  for (int i=0; i<10; i++ ) {
    printf ( "z␣%d␣=␣%lf\n", i, z);
    z = pow(z, 2) + c;
  }
}
```

Рис. 1: Пример листинга и TIKZ декорации к нему, оформленные в окружении `figure`. Обратите внимание, что рисунок отображается не там, где он в документе, а может «плавать».

### 5.2.1. Выделения куска листинга с помощью tikz

Это обывает полезно в текста, а ещё чаще — в презентациях. Пример сделан на основе вопроса на StackOveflow[5].

[5] https://tex.stackexchange.com/questions/284311 (дата доступа: 16 декабря 2022 г.).

# 6. Эксперимент (желательно к Новому году)

Как мы проверяем, что всё удачно получилось. К Новому году для промежуточного отчета желательно хотя бы описать как он будет проводиться и на чем.

## 6.1. Условия эксперимента

Железо (если актуально); версии ОС, компиляторов и параметры командной строки; почему мы выбрали именно эти тесты; входные данные, на которых проверяем наш подход, и почему мы выбрали именно их.

## 6.2. Исследовательские вопросы

По-английски называется *research questions*, в тексте можно ссылаться на них как RQ1, RQ2, и т. д. Необходимо сформулировать, чего мы хотели бы добиться работой (2 пункта будет хорошо):

- Хотим алгоритм, который лучше вот таких-то остальных.
- Если в подходе можно включать/выключать составляющие, то насколько существенно каждая составляющая влияет на улучшения.
- Если у нас строится приближение каких-то штук, то на сколько точными будут эти приближения.
- и т.п.

Иногда в работах это называют гипотезами, которые потом проверяют. Далее в тексте можно ссылаться на research questions как RQ, это общепринятое сокращение.

## 6.3. Метрики

Как мы сравниваем, что результаты двух подходов лучше или хуже:

- Производительность.
- Строчки кода.
- Как часто алгоритм «угадывает» правильную классификацию входа.

Иногда метрики вырожденные (да/нет), это не очень хорошо, но если в области исследований так принято, то ладно.

## 6.4. Результаты

Результаты понятно что такое. Тут всякие таблицы и графики, как в таблице 1. Обратите внимание, как цифры выровнены по правому краю, названия по центру, а разделители $\times$ и $\pm$ друг под другом.

Скорее всего Ваши измерения будут удовлетворять нормальному распределению, в идеале это надо проверять с помощью критерия Колмогорова и т.п. Если критерий этого не подтверждает, то у Вас что-то сильно не так с измерениями, надо проверять кэши процессора, отключать Интернет во время измерений, подкручивать среду исполнения (англ. runtime), чтобы сборка мусора не вмешивалась и т.п. Если критерий удовлетворён, то необходимо либо указать мат. ожидание и доверительный/предсказывающий интервал, либо написать, что все измерения проводились с погрешностью, например, в 5%. Замечание: если у вас получится улучшение производительности в пределах погрешности, то это обязательно вызовет вопросы.

В этом разделе надо также коснуться Research Questions.

### 6.4.1. RQ1

Пояснения

Таблица 1: Производительность какого-то алгоритма при различных разрешениях картинок (меньше — лучше), в мс., CI=0.95. За пример таблички кидаем чепчики в честь Я. Кириленко

| Resolution | TENG | LAPM | VOLL4 |
|---|---|---|---|
| 1920×1080 | $406.23 \pm 0.94$ | $134.06 \pm 0.35$ | $207.45 \pm 0.42$ |
| 1024×768 | $145.00 \pm 0.47$ | $39.68 \pm 0.10$ | $52.79 \pm 0.10$ |
| 464×848 | $70.57 \pm 0.20$ | $19.86 \pm 0.01$ | $32.75 \pm 0.04$ |
| 640×480 | $51.10 \pm 0.20$ | $14.70 \pm 0.10$ | $24.00 \pm 0.04$ |
| 160×120 | $2.40 \pm 0.02$ | $0.67 \pm 0.01$ | $0.92 \pm 0.01$ |

### 6.4.2. RQ2

Пояснения

## 6.5. Обсуждение результатов

Чуть более неформальное обсуждение, то, что сделано. Например, почему метод работает лучше остальных? Или, что делать со случаями, когда метод классифицирует вход некорректно.

| Код модуля в составе дисциплины, практики и т.п. | Трудоёмкость | Контактная работа обучающихся с преподавателем | | | | | | | | | | Самостоятельная работа | | | | | Объем активных и интерактивных |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | лекции | семинары | консультации | практические занятия | лабораторные работы | контрольные работы | коллоквиумы | текущий контроль | промежуточная аттестация | итоговая аттестация | под руководством преподавателя | в присутствии преподавателя | с использованием методических | текущий контроль | промежуточная аттестация | |
| Семестр 3 | 2 | 30 | | | | | | | | 2 | | | | 18 | | 20 | 10 |
| | | 2-42 | | | | | | | | 2-25 | | | | 1-1 | | 1-1 | |
| Итого | 2 | 30 | | | | | | | | 2 | | | | 18 | | 20 | 10 |

Таблица 2: Если таблица очень большая, то можно её изобразить на отдельной портретной странице

# 7. Применение (того, что сделано на практике)

Если применение в лоб не работает, потому что всё изложено чуть более сжато и теоретично, надо рассказать тонкости и правильный метод применения результатов. Если результаты применяются без дополнительных телодвижений, то про это можно не писать.

# 8. Угрозы нарушения корректности (опциональный)

Если основная заслуга метода, это то, что он дает лучшие цифры, то стоит сказать, где мы могли облажаться, когда

1. проводили численные замеры;
2. выбирали тестовый набор (см. *confirmation bias*)

# 9. Реализация

Очень важный раздел для будущих программных инженеров, т.е. почти для всех. Важно иметь всегда, в том числе для промежуточных отчетов по учебным практикам или ВКР.

В процессе работы можно сделать огромное количество косяков, неполный список которых ниже.

1. Реализация должна быть. На публично доступную реализацию обязательная ссылка. Если код под NDA, то об этом, во-первых, должно быть сказано явно, и, во-вторых, на защиту должны выноситься другие результаты (например, архитектура), чтобы комиссия имела возможность оценить хоть что-то.

    - Рецензент обязан оценить код (о возможности должен побеспокоиться обучающийся).

2. Код реализации должен быть написан защищающимся целиком.

    - Если проект групповой, то нужно явно выделить какие части были модифицированы защищающимся. Например, в предыдущих разделах на картинке архитектуры нужно выделить цветом то, что вы модифицировали.
    - Нельзя пускать в негрупповой проект коммиты от других людей, или людей не похожих на Вас. Например, в 2022 году защищающийся-парень делал коммиты от сценического псевдонима, который намекает на женский «гендер». (Нет, это не шутка.) На тот момент в российской культуре это выглядело странно.
    - Возможна ситуация, что вы используете конкретный ник в интернете уже лет пять, и желаете писать ВКР под этим ником на GitHub. В принципе, это допустимо (не только лишь я так считаю), но если Вы встретите преподавателя, который считает наоборот, то Вам придется грамотно отмазываться.

В Вашу пользу могут сыграть те факты, что к нику на гитхабе у Вас приписаны настоящие имя и фамилия; что в репозитории у вас видна домашка за 1й курс; и что Ваш преподаватель практики сможет подтвердить, что Вы уже несколько лет используете это ник; и т.п.

3. Если вы получаете диплом о присвоении звания программного инженера, код должен соответствовать.

   (a) Не стоит выкладывать код одним коммитом.

   (b) Лучше хоть какие-то тесты, чем совсем без них. В идеале нужно предъявлять процент покрытия кода тестами.

   (c) Лучше сделать CI, а также CD, если оно уместно в Вашем проекте.

   (d) Не стоит демонстрировать на защите, что Вам даже не пришло в голову напустить на код линтеры и т.п.

4. Если ваша реализация по сути является прохождением стандартного туториала, например, по отделению картинок кружек от котиков с помощью машинного обучения, то необходимо срочно сообщить об этом куратору на мат-мехе, иначе Государственная Экзаменационная Комиссия «порвёт Вас как Тузик грелку», поставит «единицу», а все остальные Ваши сокурсники получат оценку выше. (Это не шутка, а реальная история 2020 года.)

Если Вам предстоит защищать учебную практику, а эти рекомендации видятся как более подходящие для защиты ВКР, то ... отмаза не засчитывается, сразу учитесь делать нормально.

# Заключение

**Обязательно для промежуточного, полугодового, годового и любых других отчётов.** Кратко, что было сделано.

**Для практик/ВКР.** Также важно сделать список результатов, который будет один к одному соответствовать задачам из раздела **??**.

- Результат к задаче 1
- Результат к задаче 2
- и т.д.

Для промежуточных отчетов сюда важно записать какие задачи уже были сделаны за осенний семестр, а какие только планируется сделать.

Также сюда можно написать планы развития работы в будущем, или, если их много, выделить под это отдельную предпоследнюю главу.

# Список литературы

[1] Dahrén Martin. The usage of PCG techniques within different game genres. — 2021.

[2] What is Procedural Content Generation? Mario on the borderline / Julian Togelius, Emil Kastbjerg, David Schedl, Georgios Yannakakis. — 2011. — 06.