

分类号 \_\_\_\_\_

编号 \_\_\_\_\_

U D C \_\_\_\_\_

密级 \_\_\_\_\_



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# 本科生毕业设计（论文）

题 目： 扩展波函数坍缩算法  
至大规模内容生成

姓 名： 聂雨荷  
学 号： 11911839  
系 别： 计算机科学与工程系  
专 业： 计算机科学与技术  
指导教师： 宋轩 副教授

2023 年 6 月 1 日

CLC \_\_\_\_\_

Number \_\_\_\_\_

UDC \_\_\_\_\_

Available for reference  Yes  No



**SUSTech**

Southern University  
of Science and  
Technology

# Undergraduate Thesis

**Thesis Title: Extend Wave Function Collapse Algorithm  
to Large-Scale Content Generation**

**Student Name:** Yuhe Nie

**Student ID:** 11911839

**Department:** Computer Science and Engineering

**Program:** Computer Science and Technology

**Thesis Advisor:** Associate Professor Xuan Song

Date: June 1, 2023

# 诚信承诺书

- 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。
- 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。
- 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。
- 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

\_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

## COMMITMENT OF HONESTY

1. I solemnly promise that the paper presented comes from my independent research work under my supervisor's supervision. All statistics and images are real and reliable.
2. Except for the annotated reference, the paper contents no other published work or achievement by person or group. All people making important contributions to the study of the paper have been indicated clearly in the paper.
3. I promise that I did not plagiarize other people's research achievement or forge related data in the process of designing topic and research content.
4. If there is violation of any intellectual property right, I will take legal responsibility myself.

Signature:

Date:

# 扩展波函数坍缩算法 至大规模内容生成

聂雨荷

(计算机科学与工程系 指导教师: 宋轩)

## [摘要]:

程序内容生成 (PCG) 为游戏、电影、艺术作品等提供了便利的数字资产创作的解决方案, 现如今被广泛利用。通过 PCG, 用户可以使用较为简单的美术资产和先验的指令性输入来生成数量、规模更大的贴图、地图、关卡、角色等。

波函数坍缩算法 (WFC) 在保证原始输入的局部特征的情况下生成不同的约束组合, 用于生成设计感较强的物体。目前, WFC 已经在游戏和智慧城市的开发中得到了应用。在研究方面, WFC 与强化学习结合紧密, 能够辅助训练游戏人工智能。

然而, 目前 WFC 在时间复杂度和冲突回溯遇到了较大问题, 导致其只能在较小规模的资产上有较好的效果。为了降低时间复杂度, 本文提出了一种嵌套式的 WFC(N-WFC) 方式, 将指数级复杂度降低至多项式级别。为了减少 N-WFC 遇到冲突的回溯和不可解问题, 本文进一步提出了完整和次完成瓦片集的概念。证明了 N-WFC 与这种瓦片集结合能够生成无限的, 确定性的和非周期性的内容。从而提出了 WFC 在生成大规模, 甚至是“无限”规模资产上的解决方案。最后, 本文基于 WFC 和 N-WFC 的算法实现了无限马里奥和卡卡颂场景生成器以验证理论的有效性。

**[关键词]:** 程序内容生成; 约束满足问题; 算法优化; 游戏智能

## [ABSTRACT]:

Procedural Content Generation (PCG) provides a convenient solution for creating digital assets for games, movies, artworks, etc., and is becoming widely accepted nowadays. With PCG, users can generate a more significant number and scale of tiles, maps, levels, characters, etc., with simple art assets and several prior inputs.

Wave Function Collapse algorithm (WFC), which can guarantee the local characteristics of the original input to generate different constraint combinations, is used to generate objects with a strong sense of design. Currently, WFC has been widely used in developing games and smart cities. Regarding research, WFC is closely combined with reinforcement learning, which can assist in training game AI.

However, WFC suffers from time complexity and backtracking conflict, which makes it only work well on smaller assets. In order to reduce the time complexity, a Nested WFC (N-WFC) scheme is proposed to reduce the exponential complexity to the polynomial level. In order to reduce the backtracking and unsolvability problem of N-WFC encountering conflicts, this paper further proposes the concept of complete and sub-complete tilesets. We prove that N-WFC combined with such tile sets is able to generate infinite, deterministic, and aperiodic content. Thus, the solution that enables WFC to generate large-scale, even "infinite" scale assets is proposed. Finally, based on WFC and N-WFC algorithm, this paper implements an Infinite Mario and Carcassone scene generator to verify the theory's validity.

**[Key words]:** Program Content Generation; Constraint Satisfy Problem; Algorithm Optimization; Game Intelligence

# 目录

<b>1. 引言</b>	<b>2</b>
1.1 课题背景及意义	2
1.2 相关研究	3
1.2.1 用于场景生成的 PCG 算法	3
1.2.2 用于关卡设计的 PCG 算法	3
1.2.3 波函数坍缩算法	4
1.2.4 密铺问题	5
1.3 本文主要内容和结构	6
<b>2. 算法介绍</b>	<b>7</b>
2.1 算法简述	7
2.2 算法输入	8
2.2.1 简单瓦片模型	8
2.2.2 重叠模型	8
2.3 问题定义	8
2.3.1 瓦片集和边集	8
2.3.2 初始化	9
2.3.3 WFC 求解器	9
2.3.4 可行解	9
2.4 约束满足问题表述	10
2.4.1 算法伪代码	10

2.4.2 时间复杂度分析 . . . . .	11
2.5 算法应用 . . . . .	12
<b>3. 算法优化一：嵌套波函数坍缩 . . . . .</b>	<b>13</b>
3.1 嵌套波函数坍缩算法介绍 . . . . .	13
3.2 问题定义 . . . . .	13
3.2.1 内部 WFC . . . . .	13
3.2.2 外部生成过程 . . . . .	13
3.2.3 可行解 . . . . .	14
3.3 算法表述 . . . . .	14
3.3.1 算法伪代码 . . . . .	14
3.3.2 时间复杂度 . . . . .	15
3.4 冲突问题 . . . . .	15
<b>4. 算法优化二：完整与次完整瓦片集 . . . . .</b>	<b>16</b>
4.1 瓦片集定义 . . . . .	16
4.1.1 完整瓦片集 . . . . .	16
4.1.2 次完整瓦片集 . . . . .	16
4.2 无限性 . . . . .	17
4.2.1 定义 . . . . .	17
4.2.2 证明 . . . . .	17
4.3 非周期性 . . . . .	18
4.3.1 定义 . . . . .	18

4.3.2 证明 . . . . .	18
4.4 确定性 . . . . .	19
4.4.1 定义 . . . . .	19
4.4.2 证明 . . . . .	19
<b>5. 算法应用 . . . . .</b>	<b>20</b>
5.1 无限马里奥 . . . . .	20
5.1.1 生成合理性 . . . . .	20
5.1.2 逻辑合理性 . . . . .	21
5.1.3 关卡复杂性 . . . . .	23
5.1.4 “无限”场景 . . . . .	23
5.1.5 实现流程图 . . . . .	24
5.2 卡卡颂地图 . . . . .	25
5.2.1 次完整瓦片集 . . . . .	25
5.2.2 多样设计需求的瓦片集 . . . . .	26
5.2.3 大规模地图生成 . . . . .	26
5.2.4 无限地图生成 . . . . .	27
<b>6. 实验分析 . . . . .</b>	<b>28</b>
6.1 时间开销 . . . . .	28
6.2 关卡复杂度评价 . . . . .	29
6.3 实验设备配置 . . . . .	30
<b>7. 回顾与展望 . . . . .</b>	<b>31</b>

7.1	讨论	31
7.1.1	将 N-WFC 和次完整瓦片集扩展到 3D 场景生成	31
7.1.2	优化重叠模型	31
7.1.3	权重笔刷系统	31
7.2	总结	32
	参考文献	33
	致谢	35

本毕业设计论文的核心内容

Extend Wave Function Collapse to Large-Scale Content Generation

已被 IEEE Conference on Games 2023 接收

# 1. 引言

## 1.1 课题背景及意义

游戏的体量正在不断增长，开发者也在不断寻找创造巨大虚拟世界的新方法。程序内容生成 (PCG) 是一种不需要人工制造美术资产，就可以生成巨大而多样的游戏世界的技术。波函数坍缩算法 (Wave Function Collapse, WFC)<sup>[1]</sup>是一种流行的 PCG 算法，它可以生成具有美术特点且符合设计需求的原始输入的变体，如纹理、2D 和 3D 物体等，被广泛应用于游戏、智慧城市、强化学习等领域中<sup>[2-4]</sup>。

WFC 算法是一种约束满足问题，它提取一组原始体素集 (我们在之后将它们称为瓦片集)，并建立与之相关的一套约束规则，在一定范围内找到一组可行解满足所有的约束。然而，目前的 WFC 算法具有较高的时间复杂度。随着生成内容规模的增长，WFC 的时间复杂度呈指数级增加，生成中遇到的冲突也越来越多。因此 WFC 必须采用基于回溯搜索使算法图灵完备。这些问题进一步增大了 WFC 的性能开销，一旦涉及到大规模或是或实时无限场景生成，WFC 便无法在短时间生成可行解。

为了解决这一挑战，本课题提出了一种嵌套波函数坍缩 (Nested Wave Function Collapse, N-WFC) 算法框架，该框架使用多个内层固定大小的 WFC (Internal-Wave Function Collapse, I-WFC)，通过维持内部 WFC 之间的约束的一个外层的生成算法，将时间复杂度优化至多项式级别。为了确保 N-WFC 生成可行解，减少冲突和回溯，本课题提出了瓦片集的设计规范，引入了“完整瓦片集” (Complete Tileset) 和次完整瓦片集 (Sub-complete Tileset) 两种概念的数学定义。完整瓦片集需要大量的瓦片才能完全满足约束条件，而次完整瓦片集可以用较少的瓦片达到相似的效果，在实际开发更实用。本课题从数学上证明了两种瓦片集都可以生成无限、非周期的密铺。并且，I-WFC 在 N-WFC 框架下生成的每个结果都是确定的，不需要回溯。

本课题复现了 WFC 算法的两种实现方式和 N-WFC 算法框架，并且实际运用到了《无限马里奥》应用的无限场景生成中和一款桌面游戏《卡卡颂》<sup>[5]</sup>的大规模场景生成中。通过实验与实践数据证明本课题将大大优化了 WFC 已有的问题，使其应用于超大规模场景生成成为可能。此外，N-WFC 框架可以与启发式算法和深度学习算法良好结合，生成更加符合设计直觉的内容。

## 1.2 相关研究

### 1.2.1 用于场景生成的 PCG 算法

自 Rogue-like<sup>1</sup>游戏问世以来，场景生成中的程序内容生成 (PCG) 一直是一个热门话题。1985 年开发的《Rogue》<sup>[7]</sup>是第一款使用 PCG 的游戏。从那时起，许多用于场景生成的 PCG 算法被提出。

基于场景生成的 PCG 算法的目标是为生成过程添加随机性，使每一个玩家的体验都不同。然而，生成的场景仍应符合逻辑并迎合设计师的想法。正如游戏开发者 Herbert Wolverson<sup>[8]</sup>所解释的那样：

随机并不是为了让游戏完全随机化。它是作为种子被输入到一个算法中，该算法生成一些近似于你想要得到的东西，但确保它每次都是不同的。

早期的场景生成技术主要集中在 2D 地下城形状的游戏上。像随机房间放置 (Random Room Placement) 或者二叉空间划分 (Binary Space Partition, BSP)<sup>[9]</sup>等算法专注于创建有逻辑的，连接的房间。后来，引入元胞自动机 (Cellular Automation, CA)<sup>[10]</sup>和醉酒行走 (Drunkard Walk)<sup>[11]</sup>，使生成的场景不规则，使其看起来更像自然的不规则。沃罗诺伊图 (Voronoi Diagram)<sup>[12]</sup> 和柏林噪声 (Perlin Noise)<sup>[13]</sup>推动 PCG 产生更大的城市或大陆板块。这些算法利用随机性和噪声，并添加后处理算法，保证场景合理。这些技术适用于二维和三维自然地形的生成。然而，这些早期的算法并不适用于强设计的程序场景生成。Maxim Gumin 提出了波函数坍缩 (WFC)<sup>[1]</sup>和 MarkovJunior<sup>[14]</sup>两种算法策略，使生成结果更符合设计直觉，而不是自然的随机性。在 WFC 中，生成被描述为一个满足连通节点的二进制约束的过程。在 MarkovJunior 中，该技术基于模式匹配和约束传播。随机性不再用于生成本身，而是基于概率做出选择。

### 1.2.2 用于关卡设计的 PCG 算法

PCG 广泛用于游戏设计的各个方面，包括关卡设计。在关卡设计中它主要有两个目的。首先，它减少了创建不同关卡的负担，同时保持游戏理想的质量。其次，它

---

<sup>1</sup>Rogue-like 是一种游戏类型<sup>[6]</sup>，这种游戏具有生成随机性，即每一局游戏都会随机生成游戏内容的特点。

为玩家创造了不同的体验，鼓励他们使用内部策略来面对不可预见的情况，而不是依赖于记忆。

关卡设计中的 PCG 抽象了游戏机制，如难度节奏、情绪体验和任务流。在关卡设计中有几种 PCG 方法。构造主义和语法主义算法使用预定义的内容块，例如语法规则，或者一个接一个地随机放置块。约束驱动 (Constraint Driver) 算法<sup>[1,14]</sup>设计特定的约束并使用约束求解器寻找潜在的解。优化器 (Optimizer) 和学习器 (Learner) 算法<sup>[15-16]</sup>响应于等级特征的适应度函数和约束的设计。它们学习潜在信息并独立调整参数以满足其期望的结果。

对于辅助关卡设计的算法，无论是显式的还是隐式的，都应该暴露可控的参数和具体的决策思想<sup>[17]</sup>，供设计师使用。这些参数可以用于各种需求，如策划设计、玩家偏好和适应难度。

### 1.2.3 波函数坍缩算法

波函数坍缩 (WFC)<sup>[1]</sup>自提出以来已经获得了显著的关注，因为它生成的输出只包含与输入相似的范式 (瓦片)，这使它适合于生成具有设计目的的对象。许多游戏应用程序使用 WFC 来生成有限的场景，如 Cave of Qud<sup>[2]</sup>使用 WFC 创建的废墟，Bad North<sup>[3]</sup>使用 WFC 生成场景和维护导航网络以规划维京战争，Townscraper<sup>[4]</sup>在 WFC 基础上构建让玩家自由城镇。此外，强化学习团队 Deep Mind 使用 WFC 为他们的智能体生成竞技场<sup>[18]</sup>。

其它一些研究针对 WFC 算法进行了优化，包括将其表示为一个约束满足问题 (CSP) 以及应用 AC-3 算法进行弧一致性检查，如 Paul C. Marrell<sup>[19]</sup>所描述的。Karth<sup>[20]</sup>等人在 WFC 中研究了回溯并结合了不同的启发式算法，而 Bailly<sup>[21]</sup>等人则将遗传算法与 WFC 混合，以指导生成内容的新颖性、复杂性和安全性。Kim<sup>[22]</sup>等人提出了一种基于图的 WFC，支持 3D 场景生成中的导航。然而，WFC 算法的指数时间复杂度问题仍然存在，限制了其生成大范围内容的能力。这个问题也会导致回溯，使得它不适合商业游戏。Marian<sup>[23]</sup>试图利用 WFC 创造一款实验性游戏《无限城市》，但注意到冲突和回溯可能会导致现有部分的再生，因此提出了当前的局限性使该游戏无法真正在商业游戏得到应用。

#### 1.2.4 密铺问题

密铺，是平面图形的镶嵌问题。旨在将形状大小完全一致的几种平面图形放置在一定范围内，使得彼此之间不留缝隙，不重叠交叉的铺满整个平面。平面图形可以是规则的四边形、六边形，也可以是更加不规则的，具有艺术特征的图形。

WFC 算法可以被视为一种基于平面四边形、六边形或者是三维立方体的密铺问题。WFC 拥有的瓦片集可以被视为密铺中的平面图形组，而一个可行解则是为了在符合约束的基础上完成密铺。

王瓦片 (Wang Tile)<sup>[24]</sup>，最早由数学家王浩在 1961 年提出。它由一组方形的瓦片组成，每个瓦片的边缘都有固定的颜色，并排排列在一个矩形网格中。每个贴图的所有四条边都必须与相邻的邻居“匹配”(具有相同颜色)。王瓦片需要解决的问题是：

是否存在一个有限的二维瓦片集，在满足约束的基础上，可以无限地且非周期性地密铺整个平面。如果有的话，这个瓦片集的最小规模是多少。

王浩的学生 Berger<sup>[25]</sup>随后给出了一个满足无限非周期性密铺示例，但是这个瓦片集拥有 20426 个瓦片。他在之后的论文中将其减少到 104 个。之后，更多小型的瓦片集被计算出。Karel Culik<sup>[26]</sup>在 1996 年提出了一组只需要 13 个瓦片的王瓦片，Emmanuel Jeandel 和 Michael Rao<sup>[27]</sup>在 2015 年提出并证明了已知的最小的王瓦片集，它是仅拥有 11 个瓦片和四个颜色的瓦片集。除此之外，他们使用计算机搜索证明了 10 个瓦片或 3 种颜色不足以维持非周期性。

## 1.3 本文主要内容和结构

本文的篇章将分为以下几个部分：

**第一章：**本章为引言部分。我们首先解释了波函数坍缩算法的由来以及其应用价值。随后，我们指出波函数坍缩算法亟待解决的时间复杂度与回溯问题，并简要介绍了本课题的解决方案。我们也回顾了近几年 PCG、WFC 和密铺领域的研究状况，提出了优化 WFC 用于大规模场景生成的实用价值。

**第二章：**本章为算法介绍部分。我们进一步阐述了 WFC 算法的生成细节，随后我们回顾了波函数坍缩的两种实现模型，简单瓦片模型 (Simple-Tiled Model) 和重叠模型 (Overlapping Model) 的生成原则。之后我们严格定义了二维形式上 WFC 所求解的问题，将其表征为一种约束满足问题，给出了其与回溯算法和 AC-3 算法结合的伪代码形式，且证明了 WFC 具有指数级时间复杂度。

**第三章：**本章为算法优化部分。我们提出了嵌套的波函数坍缩算法 (N-WFC) 以优化 WFC 的时间复杂度，我们详细的介绍了算法的外层和内层的运行逻辑，维护约束的方式并给出了 N-WFC 多项式时间复杂度的证明。

**第四章：**本章为瓦片集规则介绍部分。为了解决回溯冲突问题，我们给出了完整瓦片集和次完整瓦片集的定义，并且证明了这两种瓦片集具有的三种特性 (1) 无限性 (2) 非周期性和 (3) 确定性。其中次完整瓦片集与 N-WFC 结合将完整解决 WFC 已知的问题，我们从数学上证明了优化后的 WFC 在大规模和无限内容生成上的可能。

**第五章：**本章为算法应用部分。我们将介绍使用 WFC 作为场景生成器的两个游戏《无限马里奥》和《卡卡颂》的实现逻辑。在《无限马里奥》中，除了使用 WFC 生成符合约束的解以外，我们额外考虑了关卡的可达性和复杂性。在《卡卡颂》中，我们提出了如何将 N-WFC 与启发式算法结合以更好的适应设计决策。

**第六章：**本章为实验部分。我们对第四章和第五章介绍的内容设计了对应的实验。我们验证了 N-WFC 和次完整瓦片及的时间开销优于原先 WFC。同时分析了无限马里奥的关卡复杂度。

**第七章：**本章为回顾和展望部分。我们总结并分析了本课题的工作，提出了目前工作仍有的局限性和今后的研究方向。

## 2. 算法介绍

### 2.1 算法简述

波函数坍缩算法是一种约束满足问题 (Constraint Satisfaction Problem, CSP)，它满足 CSP 算法所需的三个元素。

- **域 (Domain):** WFC 的域是一组瓦片集 (tileset)，瓦片集中有一组有限数量的，形状相同的瓦片 (tile)。为了方便，在 2D 情况下，这些瓦片通常是一些正方形，在 3D 情况下，这些瓦片通常是一些正方体。
- **约束 (Constraints):** 瓦片的四个边或者六个面具有约束。在 WFC 中，约束是二元约束。一个约束表示两个瓦片可以在一个特定方向上拼接在一起。
- **变量 (Variables):** WFC 在一个范围较大的网格 (grid) 求解，在 2D 情况下网格大小是  $M \times N$ ，在 3D 情况下是  $M \times N \times O$ 。每个格子 (cell) 是一个需要求解变量。初始情况下每个格子的状态不被确定，可以是域中任意一个瓦片。我们称这种不确定状态的格子为波 (wave)。算法以最小剩余值 (Minimal Remaining Value, MRV) 策略从网格中选取一个剩余状态最小的格子，再以随机 (Random) 策略选取一个可行的状态，将该变量赋值为一个确定的状态，并将约束传播至相邻的格子。我们称这种操作为波坍缩 (wave collapse)。

算法的执行顺序如下：

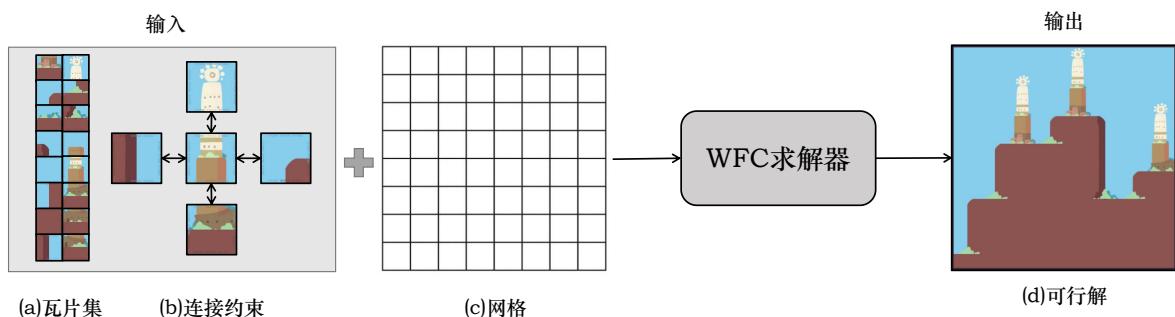


图 1 WFC 算法执行流程示意图

1. 准备一组瓦片集
2. 定义瓦片与瓦片之间的连接约束 (通常是边与边的二元约束)

3. 确定生成网格的大小，将其划分为多个格子
4. 运用 WFC 求解器，按照波函数坍缩算法的求解方式坍缩各个瓦片并将通过边约束减少邻接格子中可能的状态
5. 如果存在解，则返回一个可接受的解

## 2.2 算法输入

WFC 算法运行前需要准备一组瓦片集和相应的连接约束。WFC 为我们提供了两种实现策略，简单瓦片模型和重叠模型。两种模型仅在如何生成瓦片和连接约束上存在差异，而之后同样运用 WFC 求解器得到可行解。

### 2.2.1 简单瓦片模型

简单瓦片模型通常由手工创建。即专门绘制指定的瓦片并且人工设置需要的连接约束。这种方式非常简单且符合直觉，是大部分 WFC 应用的实现方式。

### 2.2.2 重叠模型

重叠模型可以从输入样本图像自动生成瓦片集。它利用类似卷积的方式对于输入样本图像上每个像素的局部邻域模式进行编码来提取瓦片。在 2D 情况下，我们可以指定一个  $C \times C$  的卷积核，它通过卷积的方式生成一系列瓦片。在生成的过程中，它会计算瓦片的边缘行列和溢出的一行或一列，寻找其他对边与之相同的瓦片，从而建立连接约束。

## 2.3 问题定义

在本节中，我们将完整给出 WFC 的定义，为简单起见，这个问题是在 2D 情况下表述的。

### 2.3.1 瓦片集和边集

设  $\mathcal{T}$  是一个有限的瓦片集。每一个瓦片  $t \in \mathcal{T}$  可以被表示成 4 条边的组合<sup>2</sup>

$$t = (e_n, e_s, e_w, e_e), \text{ 其中 } e_n, e_s \in \mathcal{E}_{NS}, e_w, e_e \in \mathcal{E}_{WE}.$$

$\mathcal{E}_{NS}$  和  $\mathcal{E}_{WE}$  分别为南北方向和东

---

<sup>2</sup>在 WFC 中，边的关系是唯一需要考虑的邻接约束，它是一种二元约束。

西方向的边集。在具体实现中， $\mathcal{E}_{\mathcal{WE}}$  和  $\mathcal{E}_{\mathcal{NS}}$  有可能共享也可能不会共享一些可用的边。对于一个瓦片  $t$  来说，它的每一条边可以被表示成  $e_{dir}(t)$ ,  $dir \in \{n, s, w, e\}$ 。

### 2.3.2 初始化

WFC 求解器运行在一个  $M \times N$  的网格  $\mathbf{G}$  中，其中  $g_{m,n} \subseteq \mathcal{T}$ 。我们定义在网格  $\mathbf{G}$  上的平铺函数为  $f : \mathbf{G} \leftarrow \mathcal{T}$ 。初始情况下，所有的格子都被设置成不可确定的所有状态  $\forall m \in [1, M], \forall n \in [1, N], g_{m,n} = \mathcal{T}$ 。对于每一个索引为  $(m, n)$  的格子来说，它的每一条边可以被表示为  $e_{dir}(m, n)$ ,  $m \in [1, M], n \in [1, N]$ 。

### 2.3.3 WFC 求解器

WFC 求解器按照以下顺序循环，直到完成搜索或者找到一个可行解：

1. 选择一个有最小剩余值的格子  $g_{m,n}$ 。
2. 随机选择一个可行的瓦片  $t \in g_{m,n}$ ，坍缩格子到一个确定的状态： $g_{m,n} \leftarrow \{t\}$ 。
3. 将新的约束传播到格子  $g_{m,n}$  的邻接单元中，并减少不满足约束的瓦片，直到所有的边约束被满足。

### 2.3.4 可行解

一个可行的 WFC 解被定义为<sup>3</sup>：

$$\begin{aligned} \forall_{m=1}^M \forall_{n=1}^N |g_{m,n}| &= 1 \\ e_n(m, n) &= e_s^*(m - 1, n) \\ e_s(m, n) &= e_n^*(m + 1, n) \\ e_w(m, n) &= e_e^*(m, n - 1) \\ e_e(m, n) &= e_w^*(m, n + 1) \end{aligned}$$

---

<sup>3</sup> $e_{dir}^*$  表示如果存在这样的格子，则会检查该等式。

## 2.4 约束满足问题表述

WFC 解决二元边约束问题，在生成过程中可能会产生冲突。它使用回溯算法 (Backtracking) 来确保存在可接受的解决方案，并使用 AC-3<sup>4</sup>算法来优化和保持弧一致性的同时优化算法的运行效率。

### 2.4.1 算法伪代码

算法 1、2、3 介绍了 WFC 的运行逻辑。其中算法 1 将 WFC 算法与回溯算法结合返回可行解，Algorithm 2 执行 AC-3 算法将边约束传播至相邻格子，Algorithm 3 计算对应的波坍缩。

---

#### Algorithm 1 波函数坍缩

---

**Input:** G

**Output:** 可行解，如果没有则返回 NULL

初始化所有的格子  $\forall m \in [1, M], \forall n \in [1, N], g_{m,n} = \mathcal{T}$

**while** True **do**

当所有的波都坍缩时，返回分配

$g_{m,n} \leftarrow$  最小剩余值 (G)

**for**  $\forall t \in g_{m,n}$  **do**

将  $\{g_{m,n} = t\}$  写入分配

推理  $\leftarrow$  传播 ( $g_{m,n} = \{t\}$ , G)

**if** 推理  $\neq$  False **then**

分配  $\leftarrow$  波函数坍缩 (G)

**if** 分配  $\neq$  False **then**

return 分配

**end if**

**end if**

将  $\{g_{m,n} = t\}$  从分配中移除

**end for**

**end while**

**return** NULL

---

<sup>4</sup>AC 算法会提前计算可以排除的二元约束并计算被相邻变量是否受到影响。除了 AC-3 算法外，还有 AC-4、AC-6 算法，在时间复杂度和空间复杂度上做出取舍。

---

**Algorithm 2** 传播

---

**Input:**  $g_{m,n}$ ,  $\mathbf{G}$ 创建队列  $q \leftarrow$  包含所有与  $g_{m,n}$  相邻的格子元组**while**  $q$  不为空 **do**     $(g_v, g) \leftarrow$  弹出 (queue)    **if** 波坍缩  $(g_v, g)$  **then**        将所有  $g_v$  相邻的格子对以元组的形式加入  $q$     **end if****end while**

---

**Algorithm 3** 波坍缩

---

**Input:**  $g_v, g$ **Output:** True 如果  $g_v$  有更改, 否则为 False     $\text{dir} \leftarrow g_v$  相对于  $g$  的位置,  $\text{dir} \in (n, s, w, e)$ **for**  $t \in g$  **do**     $g_{new} \leftarrow g_{new} \cup \mathcal{T}',$  其中  $\forall t' \in \mathcal{T}', f_{\overline{\text{dir}}}(t') = f_{\text{dir}}(t)$ **end for**     $g_v \leftarrow g_v \cap g_{new}$ **return** 是否变更 ( $g_v$ )

## 2.4.2 时间复杂度分析

2D 情况下, WFC 求解器运行在一个  $M \times N$  的网格  $\mathbf{G}$  中, 假设瓦片集的大小为  $|\mathcal{T}| = d$ 。则该算法有  $M \times N$  个变量, 每个变量有  $d$  个状态。回溯算法 (见 Algorithm 2) 是一种搜索算法, 它会完全搜索整个图  $\mathbf{G}$  的每一个节点 (格子), 故它的最坏时间复杂度为  $O(d^{M \times N})$ 。AC-3 算法会在搜索过程中进行剪枝, 由于每一个格子在 2D 情况下有 4 个相邻的格子, 每一个格子可以被入队至少  $d$  次, 每一次检查边的二元约束最多需要  $d^2$  次判断, 故它的时间复杂度为  $O((M \times N)^2 d^3)$ 。整体 WFC 的时间复杂度为  $O(d^{M \times N} + (M \times N)^2 d^3)$ 。它会随着网格的大小成指数级增长。

## 2.5 算法应用

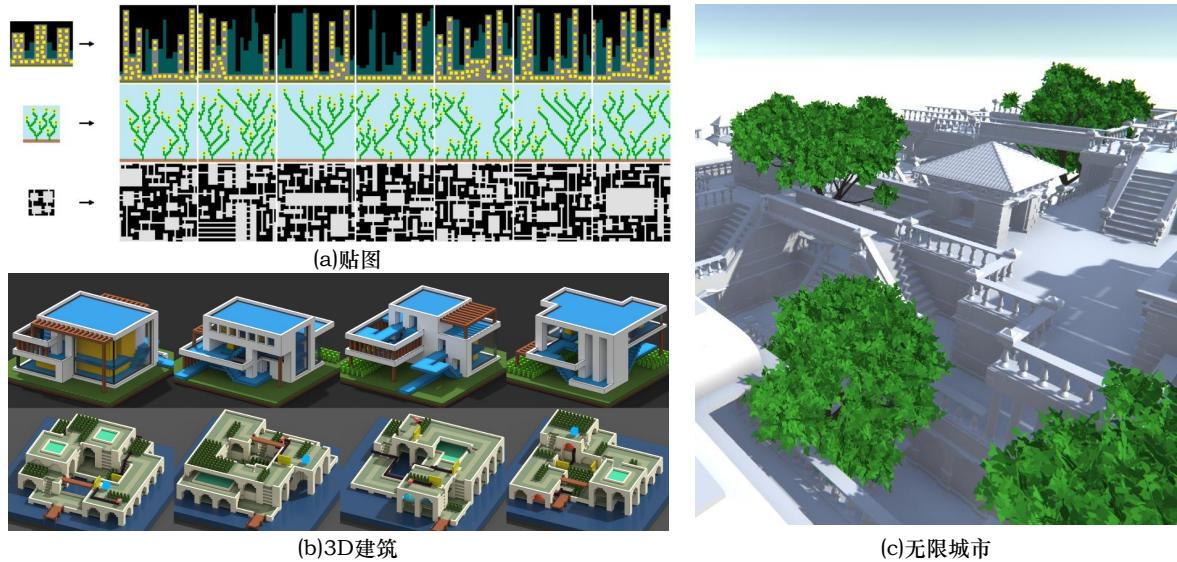


图2 WFC 算法的应用场景，包括(a)生成更大的贴图(b)生成形式多变的2D或3D的物体、建筑、场景(c)生成近似无限2D或3D场景

WFC 算法最早使用重叠模型生成 2D 贴图(如图2-a)。随后由于算法在逻辑上适用于生成 2D 或者 3D 的物体、建筑、场景等，被用于小部分游戏内容生成(如图2-b)。这之后，很多游戏使用 WFC 作为小规模场景生成的工具，比如《圣城洞》、《绝境北方》、《城镇叠叠乐》等。Marian 等人使用 WFC 探索无限内容生成，(如图2-c)。但是他指出，无限内容生成存在使用回溯算法回撤已经生成的场景，这使得已经访问的地方被重构，这些问题导致了《无线城市》暂时不适合用于商业用途。

除此之外，WFC 算法还被抽象成了音乐和诗歌生成器。比如将旋律片段抽象为一个波，片段与片段之间可以满足良好的衔接作为一个边约束，整个音乐构成一维网格。或者是将诗歌的片段抽象成一个波，片段与片段之间满足的韵脚作为一个边约束，整首诗歌构成二维网格等等。

### 3. 算法优化一：嵌套波函数坍缩

#### 3.1 嵌套波函数坍缩算法介绍

WFC 的时间开销问题给大规模内容的生成带来了挑战。为了解决这个问题，我们提出了**嵌套波函数坍缩** (Nested Wave Function Collapse, N-WFC)，它将大规模网格矩阵分割成更小的子网格 (Sub-grid)，在内部网格使用波函数坍缩的同时并保持子网格与子网格之间的约束。N-WFC 基于 CSP 的域分裂问题，在保持整体约束的前提下，将大问题分解为小任务，以达到快速计算的目的。

#### 3.2 问题定义

在本节中，我们将完整给出 N-WFC 的定义，为简单起见，这个问题是在 2D 情况下表述的。

##### 3.2.1 内部 WFC

内部 WFC (Interior WFC, I-WFC) 生成多个小的、固定大小的子网格  $\mathbf{G}^{sub} \in \mathbb{Z}^2$ ，其中，网格的大小为  $C \times C$ ， $C$  是一个常数，通常是一个很小的值。I-WFC 的工作原理类似于常规的 WFC，唯一的区别是它可能有一些从其他子网格传播的约束。具体来说， $\mathbf{G}^{sub}$  的第一行和第一列中的格子可能在开始 I-WFC 之前已经被坍缩：  
 $\exists g^{sub}, \forall m \in [1, C], |g_{m,1}^{sub}| = 1, \exists \mathbf{G}^{sub}, \forall n \in [1, C], |g_{1,n}^{sub}| = 1.$

##### 3.2.2 外部生成过程

为了生成 N-WFC 结果，我们将大规模网格矩阵分解为子网格问题，并运行 I-WFC 来解决问题。假设整个任务包含  $A \times B$  个子网格，则整个任务包含  $M \times N$  个格子，其中  $M = A \cdot (C - 1) + 1$ ,  $N = B \cdot (C - 1) + 1$ 。在后面的章节中，我们使用索引  $(a, b)$  来表示每一个子网格  $\mathbf{G}^{a,b}$ 。

这里为简单起见，外部生成过程从左上角开始，按对角线顺序逐层进行(我们将其称为对角生成过程)。在生成过程中，每个子网格包含其左邻居的最右列和上邻居的最后一行(如果存在)的值，确保相邻子网格的相邻边是一致的，可以作为同一条边进行重叠。当所有子网格生成后，每个相邻的子网格重叠其第一行和第一列，形成 N-WFC 的解。

在实际情况下，可以使用不同的外部生成过程，如外层嵌套 WFC，对角生成，顺序生成或者其它启发式算法。但是，对角生成的好处在于，对于每一个子网格来说，只需要考虑它的左边和上边的约束，不再需要考虑其它边。而对于一些特定的生成规则来说，需要维护子网格来自四个边的约束。

### 3.2.3 可行解

使用对角生成过程的 N-WFC 可行解被定义为<sup>5</sup>:

$$\begin{aligned} & \forall_{a=1}^A \forall_{b=1}^B \mathbf{G}^{a,b} \text{ is an accepted WFC} \\ & \wedge \forall_{a=1}^A \forall_{b=1}^B \forall_{m=1}^C g_{m,1}^{a,b} = (g_{m,C}^{a,b-1})^* \wedge |g_{m,1}^{a,b}| = 1 \\ & \wedge \forall_{a=1}^A \forall_{b=1}^B \forall_{n=1}^C g_{1,n}^{a,b} = (g_{C,n}^{a-1,b})^* \wedge |g_{1,n}^{a,b}| = 1 \end{aligned}$$

## 3.3 算法表述

### 3.3.1 算法伪代码

算法 4 介绍了 N-WFC 的运行逻辑。注意每一个子网格仍然使用 WFC 算法来解，并且会传入来自上方子网格和左方子网格的边约束。外层仍然遵循对角生成过程。

---

#### Algorithm 4 嵌套波函数坍缩

---

**Output:** 可行解

创建一个 WFC 网格  $\mathbf{G}$ ,  $|\mathbf{G}| = (A \cdot (C - 1) + 1) \times (B \cdot (C - 1) + 1)$   
将  $\mathbf{G}$  划分为  $A \times B$  个子网格，每个子网格的大小为  $|\mathbf{G}^{sub}| = C \times C$

**for** 对角线上的每一层 **do**

**for** 在特定层上的子网格  $\mathbf{G}^{a,b}$  **do**

初始化空的子网格  $\mathbb{G}$ ,  $|\mathbb{G}| = C \times C$

$\mathbb{G}_{:,1} \leftarrow (\mathbf{G}_{:,C}^{a,b-1})^*$

$\mathbb{G}_{1,:} \leftarrow (\mathbf{G}_{C,:}^{a-1,b})^*$

$\mathbb{G} \leftarrow \text{WFC}(\mathbb{G})$

$\mathbf{G}^{a,b} = \mathbb{G}$

将  $\mathbf{G}^{a,b}$  的结果拷贝到  $\mathbf{G}$  中，与  $\mathbf{G}^{a-1,b}$  和  $\mathbf{G}^{a,b-1}$ (如果存在的话) 重叠一条边

**end for**

**end for**

**return**  $\mathbf{G}$

---

<sup>5</sup> $(\mathbf{G}_{m,n}^{a,b})^*$  表示，如果子网格存在的话，等式将会被检查

### 3.3.2 时间复杂度

N-WFC 将整个任务划分成  $A \times B$  个子任务，其中每个子网格使用 I-WFC，并且仍然使用回溯算法和 AC-3 来确保返回可行解。每个子网格的解的时间复杂度为  $O(d^{C^2} + C^4 d^3)$ 。N-WFC 的上界包含了不超过  $\frac{M \times N}{C^2}$  个子任务。因此，整个 N-WFC 的时间复杂度为  $O(\frac{M \times N}{C^2} d^{C^2} + (M \times N) C^2 d^3)$ 。因为  $C^2$  是一个相对较小的常数，所以 N-WFC 只有指数级的时间复杂度。随着生成内容范围的增大，N-WFC 的时间开销将明显优于 WFC。

## 3.4 冲突问题

尽管 N-WFC 在理想情况下比 WFC 具有更低时间复杂度，但在实际应用中仍然面临边约束带来的冲突问题。特殊情况下，I-WFC 无法找到一个瓦片使得其满足特定的边约束。这将导致 N-WFC 无法生成可接受的解决方案，并需要外部生成过程中也使用回溯。N-WFC 在重叠模型上的效果不佳，重叠模型构建了规模较大的边集，但满足约束的瓦片很少。因此，在针对 N-WFC 和接下来的优化中，本文将不再讨论重叠模型。

## 4. 算法优化二：完整与次完整瓦片集

本章中，我们将介绍两种创建瓦片集的方法。并且证明这类瓦片集可以被用于生成非周期性、确定性、且无限的密铺内容生成。这些方法只需要有限数量的贴图，且适用于 2D 或者 3D 的瓦片。完整瓦片集可以在不产生任何冲突的情况下使用 WFC 生成，而次完整瓦片集可以使用 N-WFC 算法框架来解决冲突问题。

### 4.1 瓦片集定义

在定义瓦片集之前，为了保持非周期特性，我们需要满足  $\max\{|\mathcal{E}_{\mathcal{NS}}|, |\mathcal{E}_{\mathcal{WE}}|\} \geq 2$ ，这意味着至少一个边集包含两种或两种以上的边。

#### 4.1.1 完整瓦片集

完整瓦片集（如图3所示），被定义为：

$$\forall e_n, e_s \in \mathcal{E}_{\mathcal{NS}}, \forall e_w, e_e \in \mathcal{E}_{\mathcal{WE}}, \exists t \in \mathcal{T} \text{ s.t. } t = (e_n, e_s, e_w, e_e)$$

一个完整瓦片集的大小只少为： $|\mathcal{T}| \geq |\mathcal{E}_{\mathcal{NS}}|^2 \cdot |\mathcal{E}_{\mathcal{WE}}|^2$

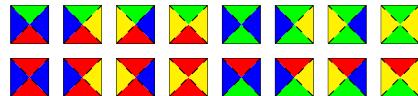


图 3 一个完整瓦片集的示例。边的颜色表示不同的边类型。 $\mathcal{E}_{\mathcal{NS}} = \{\text{红色, 绿色}\}$  和  $\mathcal{E}_{\mathcal{WE}} = \{\text{蓝色, 黄色}\}$ ，瓦片包含了所有边类型的组合，并满足完整平铺集的最小要求。 $|\mathcal{T}| = |\mathcal{E}_{\mathcal{NS}}|^2 \cdot |\mathcal{E}_{\mathcal{WE}}|^2 = 2^2 \cdot 2^2 = 16$ 。

#### 4.1.2 次完整瓦片集

次完整瓦片集（如图4所示），被定义为：

$$\begin{aligned} \forall e_n, e_s \in \mathcal{E}_{\mathcal{NS}}, \exists t \in \mathcal{T} \text{ s.t. } e_n(t) = e_n \wedge e_s(t) = e_s \\ \forall e_w, e_e \in \mathcal{E}_{\mathcal{WE}}, \exists t \in \mathcal{T} \text{ s.t. } e_w(t) = e_w \wedge e_e(t) = e_e \\ \forall e_n \in \mathcal{E}_{\mathcal{NS}}, \forall e_w \in \mathcal{E}_{\mathcal{WE}}, \exists t \in \mathcal{T} \text{ s.t. } e_n(t) = e_n \wedge e_w(t) = e_w \\ \forall e_s \in \mathcal{E}_{\mathcal{NS}}, \forall e_e \in \mathcal{E}_{\mathcal{WE}}, \exists t \in \mathcal{T} \text{ s.t. } e_s(t) = e_s \wedge e_e(t) = e_e \end{aligned}$$

一个子完整瓦片集的大小至少为： $|\mathcal{T}| \geq \max\{|\mathcal{E}_{\mathcal{NS}}|^2, |\mathcal{E}_{\mathcal{WE}}|^2\}$ 。次完整瓦片集有很多种组合。如果两瓦片集  $\mathcal{T}_1$  和  $\mathcal{T}_2$  拥有相同的边集  $\mathcal{E}_{\mathcal{NS}}$  和  $\mathcal{E}_{\mathcal{WE}}$ ， $\mathcal{T}_1$  是一个次完整瓦片

集，且  $\mathcal{T}_2$  是一个完整瓦片集的话，那么  $\mathcal{T}_1 \subseteq \mathcal{T}_2$ 。

次完整瓦片集满足完整瓦片集的所有特征，但在相同大小的边集下，次完整瓦片集所需的瓦片数量比完整瓦片集要少得多，这使其成为游戏制作中更实用的选择。



图 4 一个次完整瓦片集的示例。边的颜色表示不同的边的类型。

## 4.2 无限性

瓦片集的无限性特征表明，无论问题范围大小，算法总能返回至少一个可行解。

### 4.2.1 定义

一个可行的无限的密铺  $f : \mathbf{G} \leftarrow \mathcal{T}$  被定义为：

$$\begin{aligned} & \forall_{m=1}^{\infty} \forall_{n=1}^{\infty} |g_{m,n}| = 1 \\ & \wedge e_n(m, n) = e_s^*(m - 1, n) \wedge e_s(m, n) = e_n^*(m + 1, n) \\ & \wedge e_w(m, n) = e_s^*(m, n - 1) \wedge e_e(m, n) = e_w^*(m, n + 1) \end{aligned}$$

### 4.2.2 证明

#### (1) 完整瓦片集

对于完整瓦片集来说，很容易证明它满足无限性的特征。对于一个格子  $g_{m,n}$  来说，最严格的约束为，当它相邻的四个格子已经坍缩时，存在至少一个瓦片  $t \in \mathcal{T}$  使得其满足任意四个边约束：

$$\begin{aligned} e_n(t) &= e_s(m - 1, n) \wedge e_s(t) = e_n(m + 1, n) \wedge \\ e_w(t) &= e_e(m, n - 1) \wedge e_e(t) = e_w(m, n + 1) \end{aligned}$$

而根据完整瓦片集的定义。我们总能找到至少一个瓦片来满足这些约束。

#### (2) 次完整瓦片集

证明次完整瓦片集合上的无穷性。我们可以使用 N-WFC 中提到的对角线生成过程。假设一个“无穷”大的网格  $\mathbf{G}$  从最左上角  $\mathbf{G}^{1,1}$  开始以对角生成方式生成。在这个过程中，一个瓦片最多边约束是当它的上部和左侧的相邻约束已经被在预约束中

设置好了的时候。对于一个格子  $g_{m,n}$  来说，我们需要找到至少一个瓦片  $t \in \mathcal{T}$ ，使得：

$$e_n(t) = e_s(m-1, n) \wedge e_w(t) = e_e(m, n-1)$$

而根据次完整瓦片集的定义，我们总能找到至少一个瓦片来满足这些约束。

$$\forall e_n \in \mathcal{E}_{\mathcal{NS}}, \forall e_w \in \mathcal{E}_{\mathcal{WE}}, \exists t \in \mathcal{T} \text{ s.t. } e_n(t) = e_n \wedge e_w(t) = e_w$$

### 4.3 非周期性

瓦片集的非周期性特征确保算法生成非重复性内容，这意味着每一代都有不同的结果，使游戏内容具有 Rogue-like 的特点。

#### 4.3.1 定义

一个周期性的，且周期为  $(a, b) \in \mathbb{Z}^2$  的密铺  $f : \mathbf{G} \leftarrow \mathcal{T}$  被定义为：

$$\forall (m, n) \in \mathbb{Z}^2 : |g_{m,n}| = 1 \wedge g_{m,n} = g_{m+a, n+b}$$

$f$  被称为非周期性的如果  $f$  没有周期性。

#### 4.3.2 证明

假设在网格  $\mathbf{G}$  上仍然使用对角生成过程。考虑任意一个尚未坍缩的格子  $g_{m,n}$ ，该格子有两个相邻的格子  $g_{m,n-1}, g_{m-1,n}$ （上方和左侧）已经坍缩。注意如果  $m = 1$  或  $n = 1$ ，相应的格子可能不存在，但这不是问题，因为我们可以假设这样的相邻子网格被分配成了一个接受任意边约束的瓦片。

不论  $\mathcal{T}$  是完整还是次完整的瓦片集，根据定义，至少存在两个及以上的瓦片  $t \in \mathcal{T}$  满足两个边约束  $e_n(t) = e_s(m-1, n), e_w(t) = e_e(m, n-1)$ 。因此，对于任何一个格子  $g_{m,n}$ ，至少有两个瓦片可供选择，这样使得密铺转换成了一个独立随机过程（Independent Random Process）。这个过程适用于任何未折叠的格子，使得不论是使用完整或次完整瓦片集的密铺过程  $f$  都是非周期性密铺。

## 4.4 确定性

瓦片集的确定性特征是一个上下文定义。它是在结合使用 N-WFC 和次完整瓦片集的基础上被定义的。虽然次完整瓦片集可以直接使用细粒度对角线生成过程，而不依赖于嵌套 WFC，但 N-WFC 和子完全瓦片集的组合更适合设计目的，这将在后面的章节中得到讨论。

N-WFC 框架非常适合无限内容生成。实际上，无限意味着场景将在用户到达之前被生成好。一旦被用户观测到后，场景不会被替换或修改。《无线城市》的开发者 Marian Kleinebery 提到，当前的 WFC 存在这样一个问题，即有些地方即使已经被玩家所观测到，却因为冲突而需要回溯和重新生成。

在我看来，这种限制使得 WFC 方法不适合商业游戏。

瓦片集的确定性特征直接解决了这个问题。我们证明 N-WFC 生成的每个子网格在整个生成过程中都不会被回溯。

### 4.4.1 定义

一个使用 N-WFC 密铺  $f : \mathbf{G} \leftarrow \mathcal{T}, \mathcal{T}$  被称为确定性的，如果

$$\forall_{a=1}^A \forall_{b=1}^B f : \mathbf{G}^{a,b} \leftarrow \mathcal{T} \text{ 是一个不会被回溯的 I-WFC 可行解}$$

### 4.4.2 证明

对于 N-WFC 来说，最强的约束是  $\exists \mathbf{G}^{sub}, \forall m \in [1, C], |g_{m,1}^{sub}| = 1, \exists \mathbf{G}^{sub}, \forall n \in [1, C], |g_{1,n}^{sub}| = 1$ 。当我们把子网格  $\mathbf{G}^{sub}$  拆分成小的格子时，所有的约束都与  $e_n(t)$  和  $e_w(t)$  相关。就像在无限性的章节中证明的那样，对角生成过程在这些约束的情况下仍然能找到可行解。而实际上在 N-WFC 中，每个子网格使用 I-WFC 求解。由于 I-WFC 本身与 WFC 一样采用回溯策略，因此必然返回可行解。所以，使用子完整瓦片集的 N-WFC 是确定性的。

## 5. 算法应用

本章中，我们构建了 N-WFC 的两个应用示例。首先我们使用 WFC 和重叠模型结合一维的嵌套构造了《无限马里奥》的地图场景，并且深入介绍了为了满足游戏可玩性的其它前处理和后处理。之后我们介绍了使用 N-WFC 和次完整瓦片集的构造《卡卡颂》地图场景，概述了其与启发式算法结合辅助设计决策。

### 5.1 无限马里奥

我们使用 WFC 为 2D 横板跳跃游戏无限马里奥游戏生成地图。按照如下几个实现步骤生成：

1. **生成合理性：** 使用马里奥原始地图作为输入素材，通过 WFC 的重叠模型提取瓦片和相关约束。加上部分预处理生成输入子图(子网格)。
2. **逻辑合理性：** 为了使得生成的子图是可玩的，即至少存在一条路径使得玩家控制的人物能够通过子图生成的关卡内容，我们加入了后处理算法来检查是否存在这样的可行路径。
3. **关卡复杂性：** 为了使游戏本身具有难度递增的关系，我们使用一定的启发式算法对子图进行计算。
4. **“无限”场景：** 无限马里奥使用一维嵌套 WFC 进行生成，子图与子图之间重叠相同的一列。但由于重叠模型生成的瓦片集并不满足次完整瓦片集标准，且不能保证生成的子图具有逻辑合理性，故在嵌套过程中使用回溯算法保证可行解。

在下面的章节中，我们将一一介绍每一个步骤。

#### 5.1.1 生成合理性

首先，我们使用一张 2D 横板跳跃游戏马里奥的经典地图(图5-a)作为输入数据集。然后，我们使用 WFC 的重叠模型从中提取了 24 种的  $16 \times 16$  像素瓦片与邻接关系，如图5-b 所示，每一个瓦片旁边四周的边代表了重贴模型提取的相邻边约束。重叠模型提取瓦片四条边和它向外一个像素的四条边的关系进行计算，寻找能与之相

邻的瓦片。在图5-b中，越小的瓦片周围相邻的边约束越多。最后，我们使用 WFC 算法(图5-c)在一个  $12 \times 20$  的网格中求解出马里奥关卡的子图。

在马里奥关卡中，我们为 WFC 算法加入了预处理以确保更好的效果。从马里奥游戏中可以观察到，马里奥的上面两方格多为天空和云朵，下面两方格多为砖块。在重叠模型获取瓦片的时候，我们取前两行格子( $32 \times 32$ 个像素)标记为“天空”瓦片，后两行格子标记为“地面”瓦片。初始化网格时，前两行格子初始化中可选的瓦片只能是标记为“天空”瓦片，后两行只能是标记为“地面”瓦片(图6)。WFC 在此基础上进行坍缩。

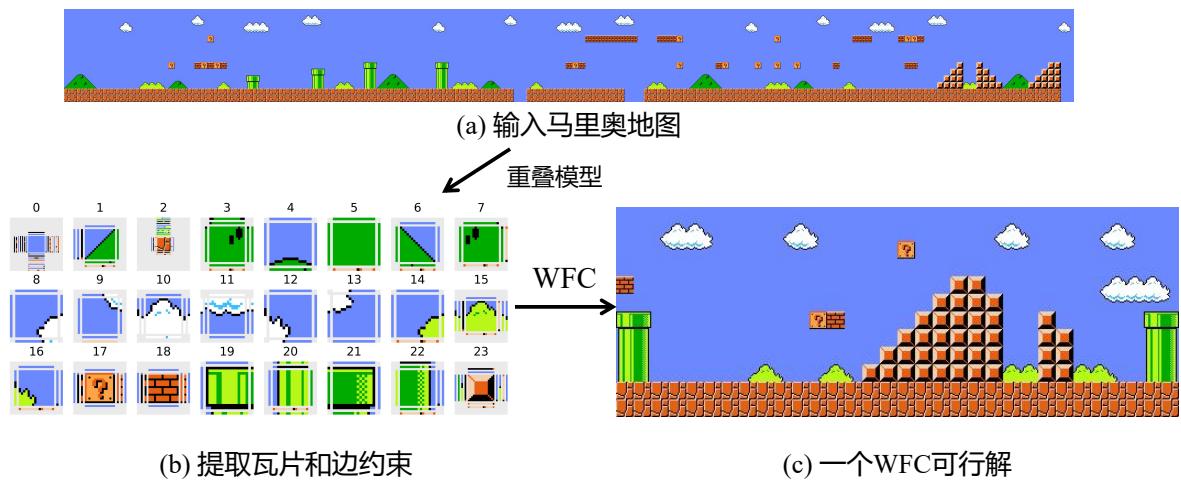


图 5 使用 WFC 重叠模型构建马里奥场景

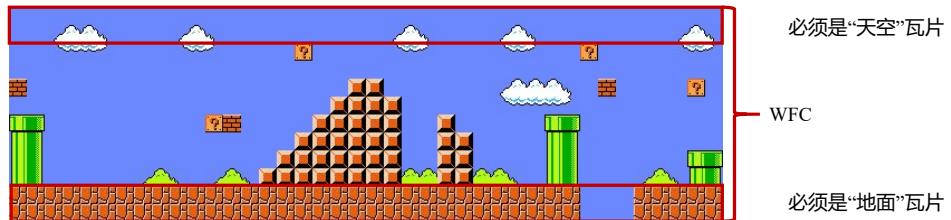


图 6 WFC 模型预处理

### 5.1.2 逻辑合理性

生成合理性使用 WFC 保证生成的子图是视觉上合理的。但是在实际将图作为关卡之前，还需要保证玩家可以操控马里奥从左边走到右边。我们认为存在至少一种路径使得玩家可以从最左边移动到最右边的图符合游戏的逻辑合理性。

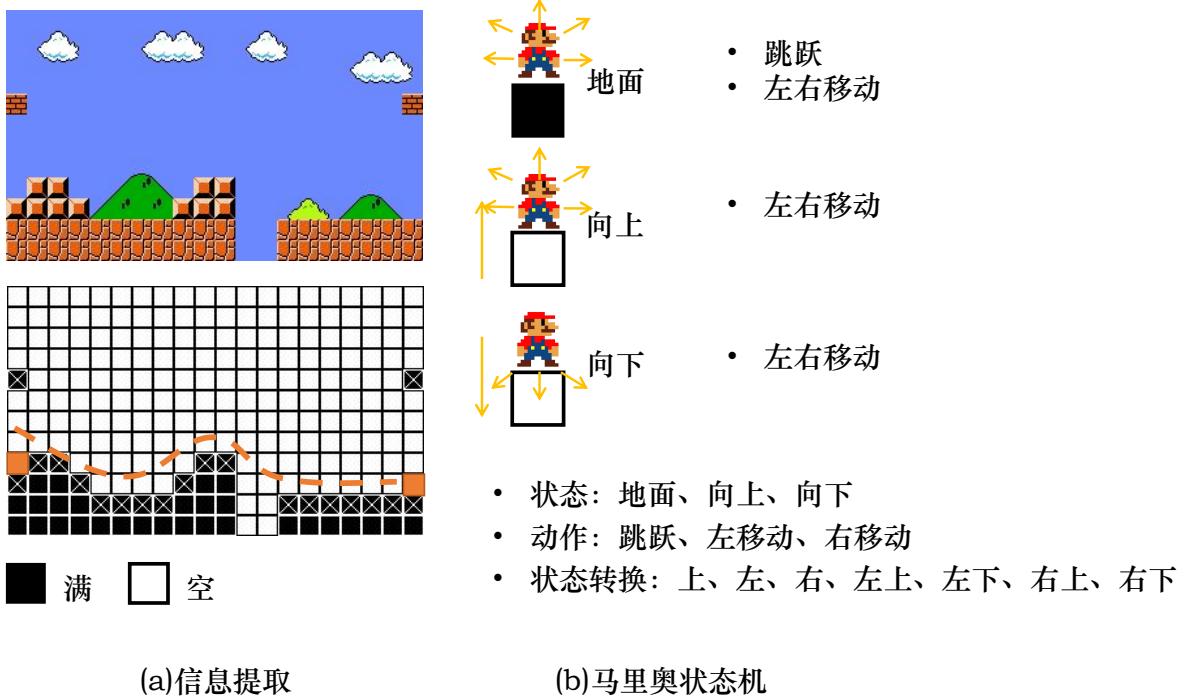


图 7 子图逻辑合理性分析与马里奥状态机

马里奥关卡的瓦片可以被分为两种类型。一种是“满”类型，角色会被这种瓦片阻挡或者可以站立在上面，一种是“空”类型，角色可以自由地通过这些地方。WFC 生成的子图网格首先会将网格中的瓦片分为不同的类型 (7-a)，整体表示为一个二维矩阵， $\text{满} = 1$ ， $\text{空} = 0$ 。

随后，我们构建马里奥角色的状态机。在游戏中，马里奥一共有三种状态 (图7-b):

1. 地面：马里奥在地面上，与“满”类型的瓦片接触的时候马里奥角色处于地面。
2. 向上：马里奥发起跳跃动作后，正在空中向上运动的一段时间内的状态。
3. 向下：马里奥在没有遇到“满”类型的瓦片前，向下运动的一段时间内的状态。

在整个游戏中，马里奥可以执行的动作有，跳跃 (对应按空格或者上方向键)；左右移动 (对应按 AD 或者左右方向键)。在不同的状态下，马里奥可以采取不同的动作。

为了验证逻辑合理性，算法从第一列从下往上数第一个为“空”的瓦片位置 A 开始，找到最后一列从下往上数第一个为“空”的瓦片位置 B 结束，试图找到一条可行的路径 (两个节点在图7-a 中以橙色方块表示)。网格的每一个位置都是一个节点，算法将尝试构建一张图数据结构。角色初始被放置在位置 A，从地面状态开始。每一个

状态下搜索角色可以执行的动作并且计算出执行动作后角色新的状态和位置，并将两个位置节点连通。使用广度优先搜索直到搜索到位置 B 的节点为止。如果能够搜索到，证明子图是逻辑合理的。

### 5.1.3 关卡复杂性

为了使生成的关卡具有更多的可玩性，我们用一些启发式的算法来计算关卡的复杂程度。这些启发式包括：

1. “满”瓦片所占百分比：占比越高的子图越复杂。因为我们粗略地认为阻挡运动的瓦片越多，角色需要采取更多的行动。
2. “空”瓦片在“地面”瓦片所占百分比：占比越低的子图越复杂。因为我们粗略地认为地面如果为空，则角色需要执行更多的跳跃操作，游戏关卡更复杂。
3. 从开始节点 A 到结束节点 B 的最短路径长度：越长则子图越复杂。最短路径表示至少需要运动的距离，我们粗略地认为当路径比较长的游戏关卡越复杂。
4. 从开始节点 A 到结束节点 B 最短路径需要执行跳跃操作数量：越多则子图越复杂。我们粗略的认为跳跃操作越多的游戏关卡越复杂。

### 5.1.4 “无限”场景

为了实现马里奥场景的无限效果，我们使用一维的 N-WFC，即重叠子网格与子网格之间的一列。这是因为马里奥是一款横板 2D 游戏，它的地图是不断向右延伸的，我们只需要保证每一个子图的最右列与新的子图的最左列一致即可。

由于基于重叠模型生成的瓦片和约束并不满足我们之前定义的次完整瓦片集的要求，这也就意味着在将右边一列作为约束传播到下一张子图时，存在冲突的可能性。除此之外，生成的子图也可能不满足我们之前提到的逻辑合理性。这两种情况都会导致无法生成满足要求的子图，所以在无限马里奥中，我们在 N-WFC 的外部生成过程中加入了回溯算法。



图 8 使用一维的 N-WFC 实现无限马里奥效果

### 5.1.5 实现流程图

无限马里奥有两种具体的实现方式。一种方式是预先生成好一定数量的子图，游戏运行中请求指定索引的子图。另一种方式是当玩家操控的角色运行到一定位置时，再请求生成新的子图。下面的流程图(图9)主要介绍第一种生成方式。图10展示了一个无限马里奥的生成示例。

1. 当生成的子图数量小于所需的数量时 ( $\text{image} < \text{IMG\_NUM}$ )，算法继循环执行。
2. 当算法因为逻辑合理性回溯的次数超过一定范围时，将会回溯上一张子图 ( $\text{failure connectiveness} < \text{CONN\_MAX}$ )。
3. 将上一张子图的最右侧一列传入 WFC 中生成新的子图，如果无法生成则回溯上一张子图。
4. 检查该子图的逻辑合理性，如果不连通的话回到第二步 ( $\text{connectiveness}(\text{image})$ )。
5. 记录该子图的关卡复杂性，可以设置回溯不符合难度的子图 ( $\text{difficulty}(\text{image})$ )。
6. 保存子图。
7. 当生成的子图数量等于所需的数量时，将其按顺序拼成一张无限马里奥关卡地图并返回。

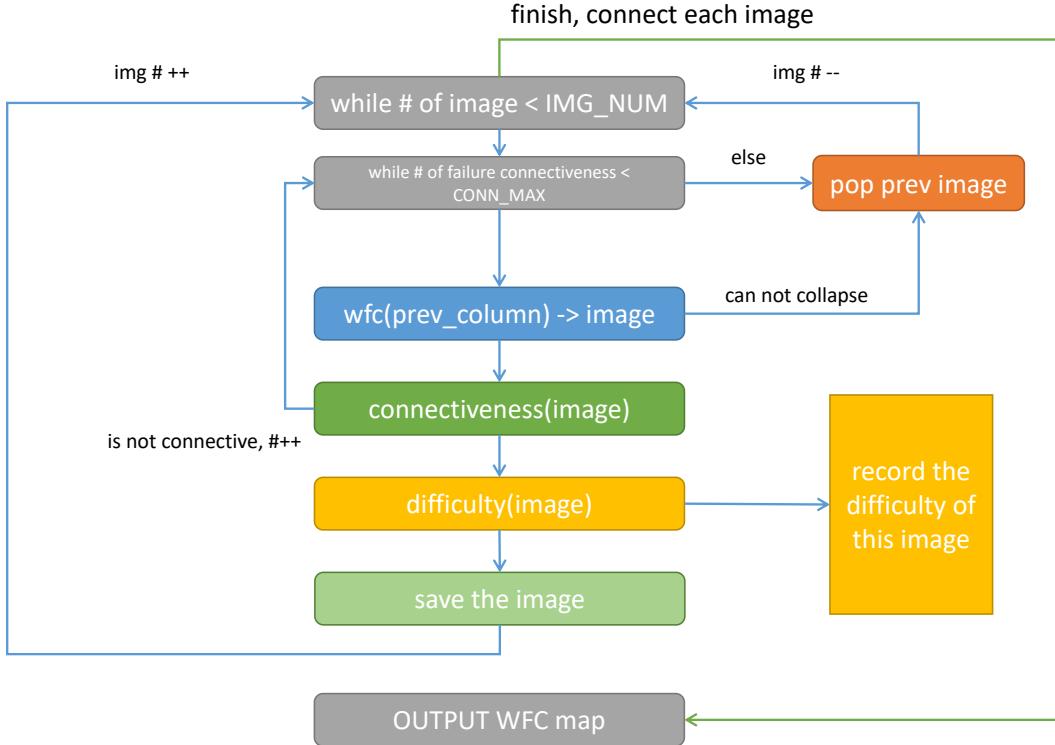


图 9 无限马里奥算法流程图



图 10 无限马里奥实现效果

## 5.2 卡卡颂地图

我们使用桌游《卡卡颂》(Carcassonne)作为生成确定性大规模或无限内容生成的示例。卡卡颂的原先规则是玩家轮流抽取瓦片构建出一个中世纪的地图。且必须保证手上的瓦片能够衔接地图的相邻地形(也就是满足边的约束)。玩家通过抢占地形来获得分数。在下面的实现中，我们不再沿用这个游戏规则。我们主要侧重于如何使用卡卡颂的瓦片集生成地图，并且能够使用一定的启发式规则来约束卡卡颂的生成效果。

### 5.2.1 次完整瓦片集

卡卡颂在逻辑上提供了四种边类型： $e_0$ -草边缘， $e_1$ -城市边缘， $e_2$ -路径边缘和 $e_3$ -溪流边缘。在这个游戏中， $\mathcal{E}_{\mathcal{NS}} = \mathcal{E}_{\mathcal{WE}} = \{e_0, e_1, e_2, e_3\}$ 。据此，我们创建一个满足次完整瓦片集(图11)，它包含28个瓦片。需要注意的是我们使用的次完整瓦片集是一个非常小的子集。在游戏的实际实现中，瓦片可以是对称的，也可以根据需要旋转。具

有相同逻辑形式的多个瓦片也被接受，提供了更多的组合情况。

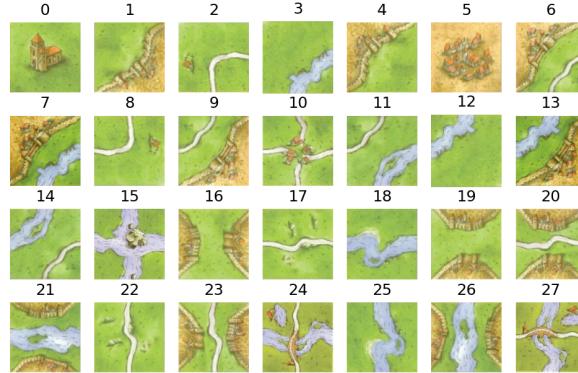


图 11 一个卡卡颂次完整瓦片集示例

### 5.2.2 多样设计需求的瓦片集

在第 4 章中，我们讨论了瓦片集在逻辑上的形式。但是，具有相同逻辑形式的瓦片可以具有不同的设计属性，从而体现游戏本身多样的艺术效果。如图 12 所示，(a)-(d) 是相同的瓦片  $t = (e_0, e_0, e_0, e_0)$ ，但具有不同的艺术表达。在设计中，我们可以把瓦 (a) 想象成一个普通的房子，把瓦 (d) 想象成一个豪宅。这和瓦片 (e) 比瓦片 (h) 更豪华是一样的。标签“豪华感”是一种设计属性，它可以运用在逻辑相同但是设计不同的瓦片上，使得生成的地图更加符合设计需求。

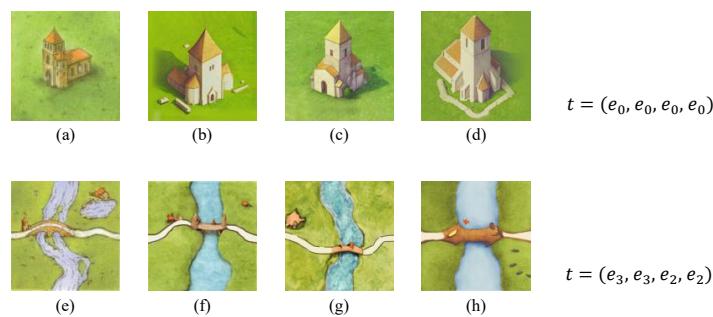


图 12 逻辑上相同但设计属性上不同的瓦片具有不同的设计属性（瓦片 (a) 和 (e) 是 Carcassonne 的原始瓦片，而其它的则由 DALL·E 生成，它们在瓦片的定义上是一样的，却有不同的效果）

### 5.2.3 大规模地图生成

我们使用 N-WFC 生成卡卡颂的大规模地图，我们首先设置 N-WFC 生成的网格的大小 (图13-b) 并按对角生成的方式将网格相邻网格的约束传递给 I-WFC 生成 (图13-c)。生成所有子网格时，每个子网格与另一个相邻的子网格共享一条边，并重

叠形成最终结果(图13-d)。

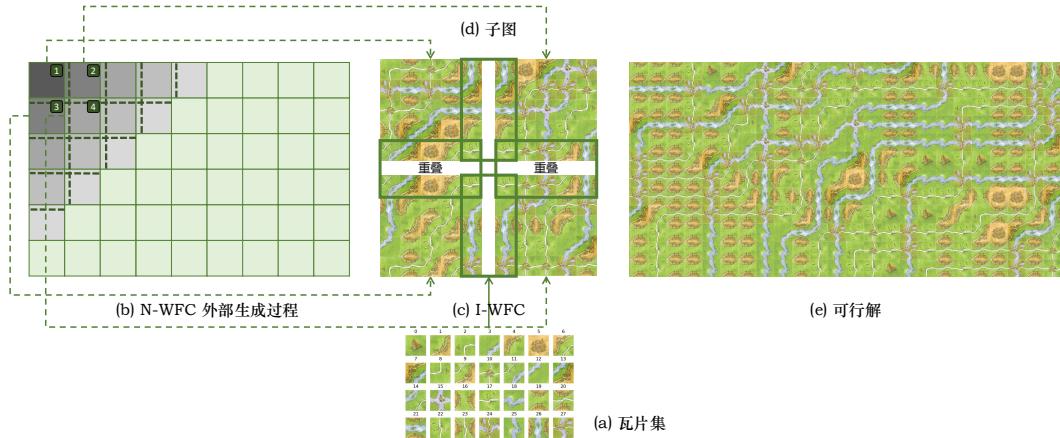


图 13 使用 N-WFC 实现卡卡颂地图生成示例

#### 5.2.4 无限地图生成

为了生成无限的地图，我们修改了外部生成过程，以玩家为中心，预生成 I-WFC 最接近的九个子网格(图 14-b)。当玩家移动到不同的位置时，N-WFC 会检测以玩家所在的新位置为中心的九个相邻的子网格中是否存在任何未生成的子网格，如果有，使用 I-WFC 持续生成新的子网格(图14-c)。

使用次完整瓦片集，I-WFC 保证能够从之前生成的子网格传播的两个相邻的行和列约束的情况下返回新的子网格的可行解。生成过程的确定性特征确保了玩家所访问的所有子网格都不会被回溯。因为 I-WFC 是在一个较小的网格上生成的，所以它在一个极短的时间内完成了生成过程，在视觉上为玩家提供了一个无限场景效果。

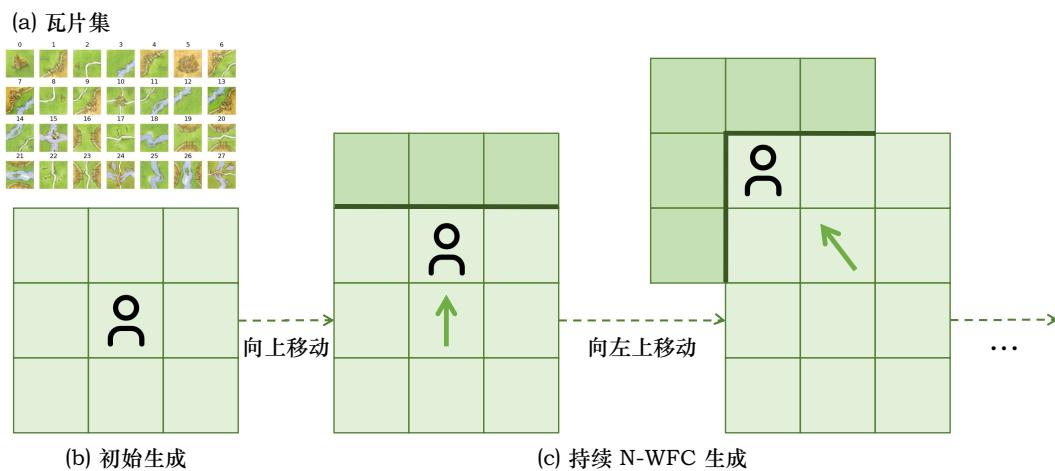


图 14 使用 N-WFC 和次完整瓦片集实现的卡卡颂地图示意图

## 6. 实验分析

### 6.1 时间开销

为了验证第3章和第4章中介绍的优化效果，我们比较了传统的 WFC 算法和将次完整瓦片集和 N-WFC 框架结合起来的性能开销。两种方法均采用最小剩余值策略来选择一个坍缩的波，并采用随机选择策略进行波坍缩。在 N-WFC 的外部，我们使用对角线生成过程。

我们创建了一系列具有不同边集的次完整瓦片集。我们为 I-WFC 赋值一个小常数  $C$ ，并评估了不同规模的网格的下时间复杂度。在实验中，我们使用了不同的瓦片集，在一系列不同规模的网格下的时间开销。我们对比了使用 N-WFC 策略和 WFC 策略的执行效果。每个实验执行了 100 次，并计算花费时间的平均值和方差。

图 15 展示了时间开销实验的结果。我们创建了 6 组有不同大小的边集的次完整瓦片集。为了简化，我们设置边集  $|\mathcal{E}_{\mathcal{NS}}| = |\mathcal{E}_{\mathcal{WE}}| = \{2, 3, 4, 5, 6, 7\}$ ，并且我们赋值  $C = 5$ 。我们通过不同的网格大小来评估 N-WFC 和 WFC 的性能  $[(5 \times 9), (9 \times 17), (17 \times 33), (25 \times 49), (33 \times 75), (41 \times 81), (49 \times 97)]$ 。

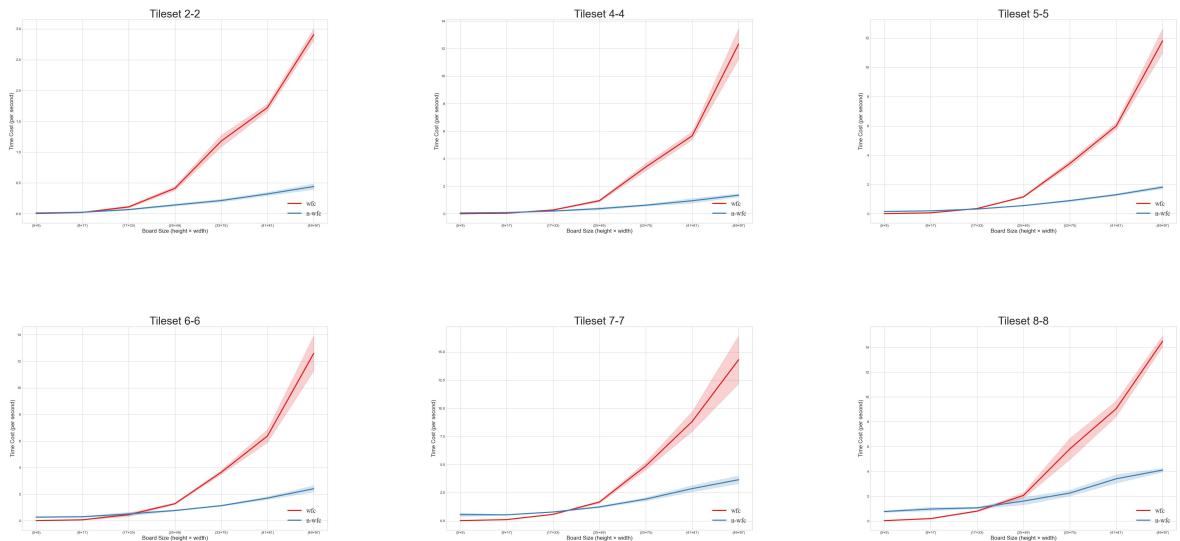


图 15 WFC 与 N-WFC 在不同次完整瓦片集中的时间开销比较。其中蓝色的线表示 N-WFC 随着网格增大的时间开销，红色的线表示 WFC 随着网格增大的时间开销。

结果表明，WFC 的时间复杂度呈指数增长，而 N-WFC 的时间复杂度呈多项式增长。对于大规模内容生成，N-WFC 仅使用 I-WFC 回溯固定范围的子网格，其时间开销显著优于传统的 WFC。

## 6.2 关卡复杂度评价

在 5.1.3 章中我们介绍了评价马里奥关卡效果的四个标准。本实验一共分析了 1000 张子图，如图 16 所示，fullness, ground fullness, path length 和 jump number 分别对应四种评价标准。其中，前两个标准的取值范围为  $[0, 1]$ ，后两个标准的取值范围是大于 0 的正整数。对于每一个标准，我们计算了它的上四分位值、平均值、下四分位值。

从图 16 中可以看到，整体上生成子图中“满”瓦片的占比在 20%-25%，而“地面”部分“满”瓦片的占比则到了 85%-100%。最短路径长度的变化范围较大，主要集中在 26-31 之间。相比较而言，跳跃次数一般只有 3-5 次。

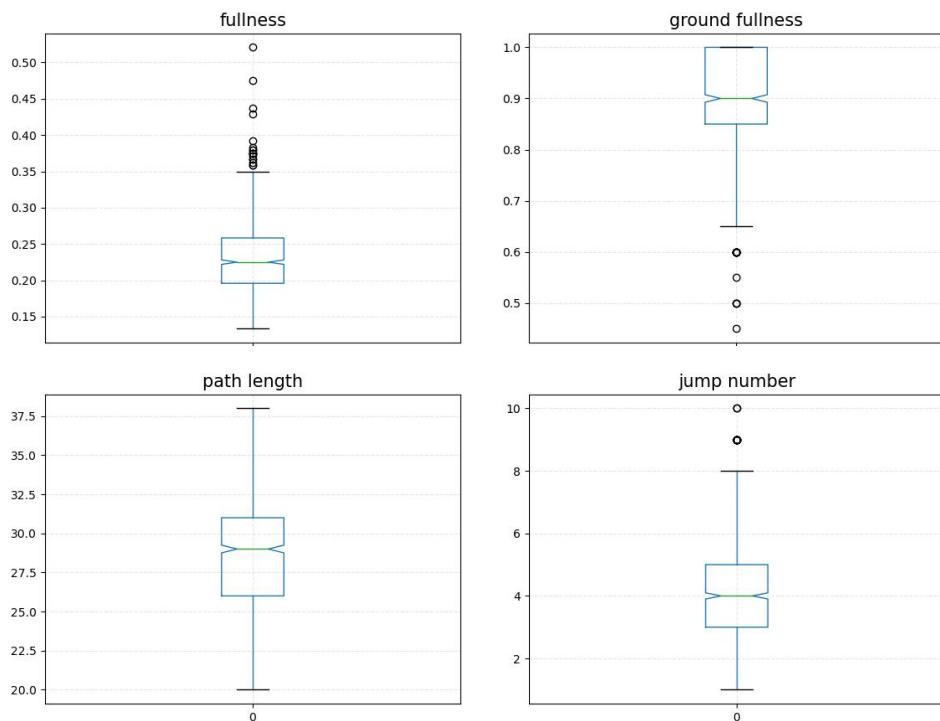


图 16 无限马里奥中关卡复杂度评价的四个指标数据分布

对于每一个子图来说，我们按照如图 17 的方法计算它的复杂度。首先我们会分别计算子图落入四种评价标准的范围。在不同范围内的评分为 0.25, 0.5, 0.75 和 1。当得出每一种标准的评分后，我们按照相同的权重进行加权求和。总评分的取值范围为  $[0,1]$ 。

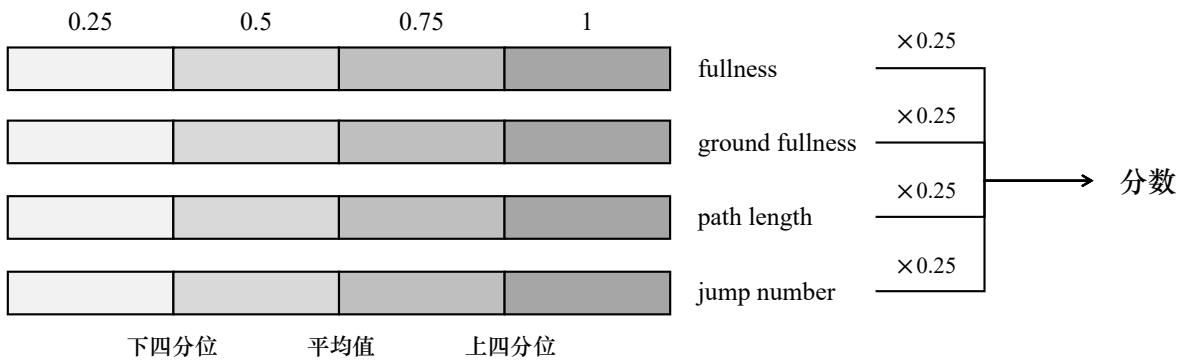


图 17 关卡复杂度计算方法

由于每一张子图有一个总评分，我们仍按照划分评价标准的方式划分成四个复杂分段。每一个分段代表不同的难度。图 18 展示了不同复杂度分段下的子图效果。可以明显看出，难度 1 的子图明显比难度 4 的子图更简单。

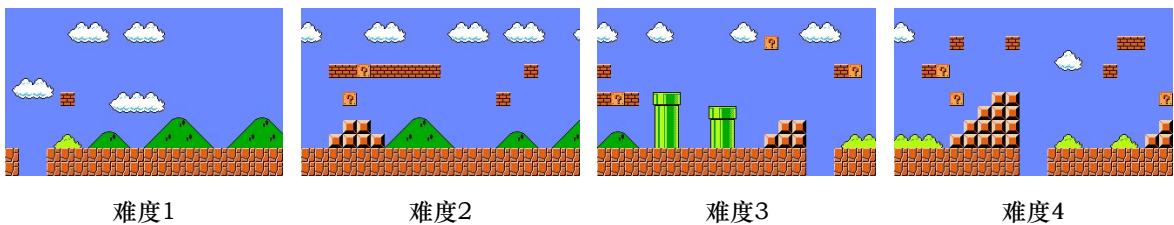


图 18 关卡复杂性评价的四个难度子图效果

### 6.3 实验设备配置

本文的所有实验均在拥有 Microsoft Windows 11 Pro 操作系统、Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz、16GB RAM 的计算机上进行。

## 7. 回顾与展望

### 7.1 讨论

#### 7.1.1 将 N-WFC 和次完整瓦片集扩展到 3D 场景生成

在前面的章节中，我们专注于在 2D 场景上的实现策略，但该 N-WFC 算法框架可以很容易地扩展到 3D。一个 3D 瓦片（也叫体块）由六个面组成。被表示为  $t = (e_n, e_s, e_w, e_e, e_u, e_d)$ ，它们分别属于不同的面集  $\mathcal{E}_{\mathcal{NS}}, \mathcal{E}_{\mathcal{WE}}, \mathcal{E}_{\mathcal{UD}}$ 。N-WFC 在一个 3D 的网格中运行  $\mathbf{G} \subseteq \mathbb{Z}^3$ ，其中包含  $M \times N \times O$  个瓦片。在 3D 的情况下，N-WFC 使用固定大小为  $C \times C \times C$  I-WFC 生成子网格。N-WFC 的时间复杂度为  $O(\frac{M \times N \times W}{C^3} d^{C^3} + (M \times N \times W) C^3 d^3)$ 。

在次完整瓦片集中，我们将可以添加五个新的约束。

$$\begin{aligned} & \forall e_u, e_d \in \mathcal{E}_{\mathcal{UD}}, \exists t \in \mathcal{T} \text{ s.t. } e_u(t) = e_u \wedge e_d(t) = e_d \\ & \forall e_n \in \mathcal{E}_{\mathcal{NS}}, \forall e_u \in \mathcal{E}_{\mathcal{UD}}, \exists t \in \mathcal{T} \text{ s.t. } e_n(t) = e_n \wedge e_u(t) = e_u \\ & \forall e_s \in \mathcal{E}_{\mathcal{NS}}, \forall e_d \in \mathcal{E}_{\mathcal{UD}}, \exists t \in \mathcal{T} \text{ s.t. } e_s(t) = e_s \wedge e_d(t) = e_d \\ & \forall e_e \in \mathcal{E}_{\mathcal{WE}}, \forall e_u \in \mathcal{E}_{\mathcal{UD}}, \exists t \in \mathcal{T} \text{ s.t. } e_e(t) = e_e \wedge e_u(t) = e_u \\ & \forall e_w \in \mathcal{E}_{\mathcal{WE}}, \forall e_d \in \mathcal{E}_{\mathcal{UD}}, \exists t \in \mathcal{T} \text{ s.t. } e_w(t) = e_w \wedge e_d(t) = e_d \end{aligned}$$

在 3D 的情况下，也很容易证明子完整瓦片集也满足之前讨论的三个特征。N-WFC 的外部生成过程可以使用不同算法来生成符合设计期望的场景。

#### 7.1.2 优化重叠模型

虽然我们已经证明了使用子完整瓦片集的简单瓦片模型的 N-WFC 可以在短时间内生成确定的无限内容，但重叠模型具有从现有图片或模型中自动提取约束关系的优势。然而，该方法目前仍不能满足子完整瓦片集的约束条件。它创建了很大的边集，但无法提取足够多的满足子完整瓦片集定义的瓦片。这导致了 N-WFC 生成过程中的大量冲突。未来的工作可以考虑修改重叠模型并进行后处理以满足要求。

#### 7.1.3 权重笔刷系统

瓦片集具有的设计属性使得 N-WFC 框架可以增加更多适配设计决策的算法。权重笔刷系统由不同的加权刷组成，每个加权刷代表带有特定设计属性标签的贴图，如

建筑、景观或任务点。我们在这里简要讨论一种可行的权重笔刷系统。游戏设计师可以自定义画笔的权重，并使用它们来绘制 N-WFC 生成的粗略场景。权重影响折叠格子的选择和折叠瓦片的选择。此外，开发者可以改变传统 WFC 算法的启发式策略。

我们认为权重笔刷系统比对角线生成过程更适合生成子网格。如果 I-WFC 使用对角线生成过程以固定顺序折叠单元格。它防止与选择策略相关的设计算法被应用。相比之下，权重笔刷系统可以与这些算法相结合。它强调了游戏设计师的作用，有助于制作出具有更好设计感的地图。

## 7.2 总结

本文围绕着一种程序化内容生成算法——波函数坍缩算法 (WFC) 详细地展开了讨论。我们首先定义了任务，介绍算法的计算方式以及分析时间复杂度。进而我们发现原始的 WFC 具有指数级的时间复杂度，使得它不适用于大规模的内容生成。于是本文提出了一种嵌套 WFC(N-WFC) 算法的优化策略。N-WFC 在保持边约束的同时，将一个内部 WFC(I-WFC) 嵌入到一个更大的外部生成过程中，将指数复杂度降低为多项式复杂度。在 N-WFC 生成过程中，我们又提出了保持边约束的完整和子完整瓦片集的概念，并证明了在这种瓦片集下，N-WFC 可以生成确定的、非周期的、无限的内容。随后我们以《无限马里奥》和《卡卡颂地图》为例探讨了两种 N-WFC 算法框架的具体实现场景和它对设计决策的帮助。最后，我们通过实验证明了 N-WFC 和次完整瓦片集组合应用能够满足预期的复杂度，分析了《无限马里奥》的关卡复杂度以及深入分析了 N-WFC 权重笔刷的效果。本文的工作解决了 WFC 在大规模和无限内容生成方面的回溯和冲突缺点，满足了游戏设计师的需求，具有可扩展性和健壮性。

## 参考文献

- [1] GUMIN M. Wave Function Collapse Algorithm[M]. 2016.
- [2] GAMES F. Caves of Qud[M/OL]. 2014. <https://www.cavesofqud.com/>.
- [3] <https://www.badnorth.com>.
- [4] STÅLBERG O. Townscraper[M/OL]. Raw Fury, 2021. [https://store.steampowered.com/app/1291340/%5C\\_/](https://store.steampowered.com/app/1291340/%5C_/).
- [5] 2023. [https://en.wikipedia.org/w/index.php?title=Carcassonne%20\(board%20game\)](https://en.wikipedia.org/w/index.php?title=Carcassonne%20(board%20game)).
- [6] Wikipedia. 2023. <https://en.wikipedia.org/w/index.php?title=Roguelike>.
- [7] Epyx. Rogue[M/OL]. 1985. <https://store.steampowered.com/app/1443430/Rogue/>.
- [8] WOLVERSON H. PRoguelike Tutorial - in Rust[M]. GitHub, 2022.
- [9] BARON J R. Procedural dungeon generation analysis and adaptation[M]. 2017: 168-171.
- [10] JOHNSON L, YANNAKAKIS G N, TOGELIUS J. Cellular automata for real-time generation of infinite cave levels[M/OL]. Monterey California: ACM, 2010: 1-4. <https://dl.acm.org/doi/10.1145/1814256.1814266>. DOI: 10.1145/1814256.1814266.
- [11] KOESNAEDI A, ISTIONO W. Implementation Drunkard's Walk Algorithm to Generate Random Level in Roguelike Games: vol. 5[M]. 2022: 97-103.
- [12] AURENHAMMER F, KLEIN R. Voronoi Diagrams: vol. 5[M]. 2000: 201-290.
- [13] HART J C. Perlin noise pixel shaders[M]. 2001: 87-94.
- [14] GUMIN M. MarkovJunior, a probabilistic programming language based on pattern matching and constraint propagation[M/OL]. 2022. <https://github.com/mxgmn/MarkovJunior>.
- [15] MOURATO F, DOS SANTOS M P, BIRRA F. Automatic level generation for platform videogames using genetic algorithms[M/OL]. Lisbon Portugal: ACM, 2011: 1-8. <https://dl.acm.org/doi/10.1145/2071423.2071433>. DOI: 10.1145/2071423.2071433.
- [16] KHALIFA A, BONTRAGER P, EARLE S, et al. PCGRL: Procedural Content Generation via Reinforcement Learning: vol. 16[M/OL]. 2020: 95-101. <https://ojs.aaai.org/index.php/AIIDE/article/view/7416>. DOI: 10.1609/aiide.v16i1.7416.
- [17] SMITH G. Techniques for AI-Driven Experience Management in Interactive Narratives[M/OL]. A K Peters/CRC Press, 2015: 552-563. <https://www.taylorfrancis.com/books/9781482254808/chapters/10.1201/b18373-48>. DOI: 10.1201/b18373-48.

- [18] TEAM O E L, STOOKE A, MAHAJAN A, et al. Open-ended learning leads to generally capable agents[M]. 2021.
- [19] MERRELL P. Comparing Model Synthesis and Wave Function Collapse[M/OL]. 2021. <https://paulmerrell.org/wp-content/uploads/2021/07/comparison.pdf>.
- [20] KARTH I, SMITH A M. WaveFunctionCollapse: Content Generation via Constraint Solving and Machine Learning: vol. 14[M/OL]. 2022: 364-376. <https://ieeexplore.ieee.org/document/9421370/>. DOI: 10.1109/TG.2021.3076368.
- [21] BAILLY R, LEVIEUX G. Genetic-WFC: Extending Wave Function Collapse With Genetic Search[M]. IEEE, 2022.
- [22] KIM H, HAHN T, KIM S, et al. Graph Based Wave Function Collapse Algorithm for Procedural Content Generation in Games: vol. 103[M]. The Institute of Electronics, Information, 2020: 1901-1910.
- [23] KLEINEBERG M. <https://mariann42.itch.io/wfc>.
- [24] WANG H. Proving Theorems by Pattern Recognition - II: vol. 40[M/OL]. 1961: 1-41. <https://ieeexplore.ieee.org/document/6773658>. DOI: 10.1002/j.1538-7305.1961.tb03975.x.
- [25] BERGER R. The undecidability of the domino problem: vol. 0[M/OL]. 1966: 0-0. <http://www.ams.org/memo/0066>. DOI: 10.1090/memo/0066.
- [26] KARI J. A small aperiodic set of Wang tiles: vol. 160[M/OL]. 1996: 259-264. <https://linkinghub.elsevier.com/retrieve/pii/0012365X9500120L>. DOI: 10.1016/0012-365X(95)00120-L.
- [27] JEANDEL E, RAO M. An aperiodic set of 11 Wang tiles[M/OL]. 2021. <https://www.advancesincombinatorics.com/article/18614-an-aperiodic-set-of-11-wang-tiles>. DOI: 10.19086/aic.18614.

## 致谢

光阴似箭，时光荏苒，大学四年即将结束。回首四年，我想感谢四年间给予我指导、帮助、关心和支持的学校、院系、老师、同学、朋友、家人和自己。

感谢南方科技大学提供的优质环境。学校教学环境和设施是如此优渥，同时又给学生丰富的奖学金和海外交流的机会。同时，我也感谢南科大为我提供的英语环境的培养，是你们让我真正意识到了英语在学术路上的关键性。

感谢计算机系对我的培养。在这里学习的大量的计算机知识，让我能够通过代码尽情发挥我的所思所想，在计算机的世界里将逻辑、理论与实践的结合，搭建出任何我想要的程序。是你们夯实了这四年我的知识储备，培养我形成会思考、会研究、会实现的好习惯。学习计算机的日子有喜有累，但总的来讲，收获满满。

感谢老师们对我的帮助。四年间，学术导师宋轩老师给予了我学术的自由，对我的学业也倍加关心。昆山杜克大学的佟馨老师带我进入了人机交互领域，给予了我第一次正式的科研机会。网易互娱的吴昊导师鼓励我参加开发的各个阶段，让我自由发挥自己的能力。除此之外，还想感谢系里的各位老师在课程上的用心负责。是你们让我四年间的能力突飞猛进。

感谢身边重要的朋友、家人们。在四年间，我遇到了三位亲切的舍友，遇到了系里的有着不同爱好和擅长不同方向的同学，在海外交流遇到了帮助我适应环境的朋友等等。感谢我的父母在我沮丧和焦虑的时候给我的鼓励与支持。与你们的相处和生活给予了我极大的动力，给每天都带来了快乐和成长。

值此毕业论文完结之际，特别感谢庄湛同学，是在大学四年间作为学长和男朋友带领我进入计算机领域，手把手教会我各种各样的知识。特别感谢郑少铭老师，对本毕业论文加以指导，提出了大量具体可行的修改意见。

最后，最想感谢的是我自己，从进校开始就一腔热血投入了计算机科学领域，并在大学四年见明确了自己在游戏智能和人机交互领域方向的道路，并一直脚踏实地、勤勤恳恳地走了下来。点滴经历都将铭记于心。谨以此论文，献给本科四年。