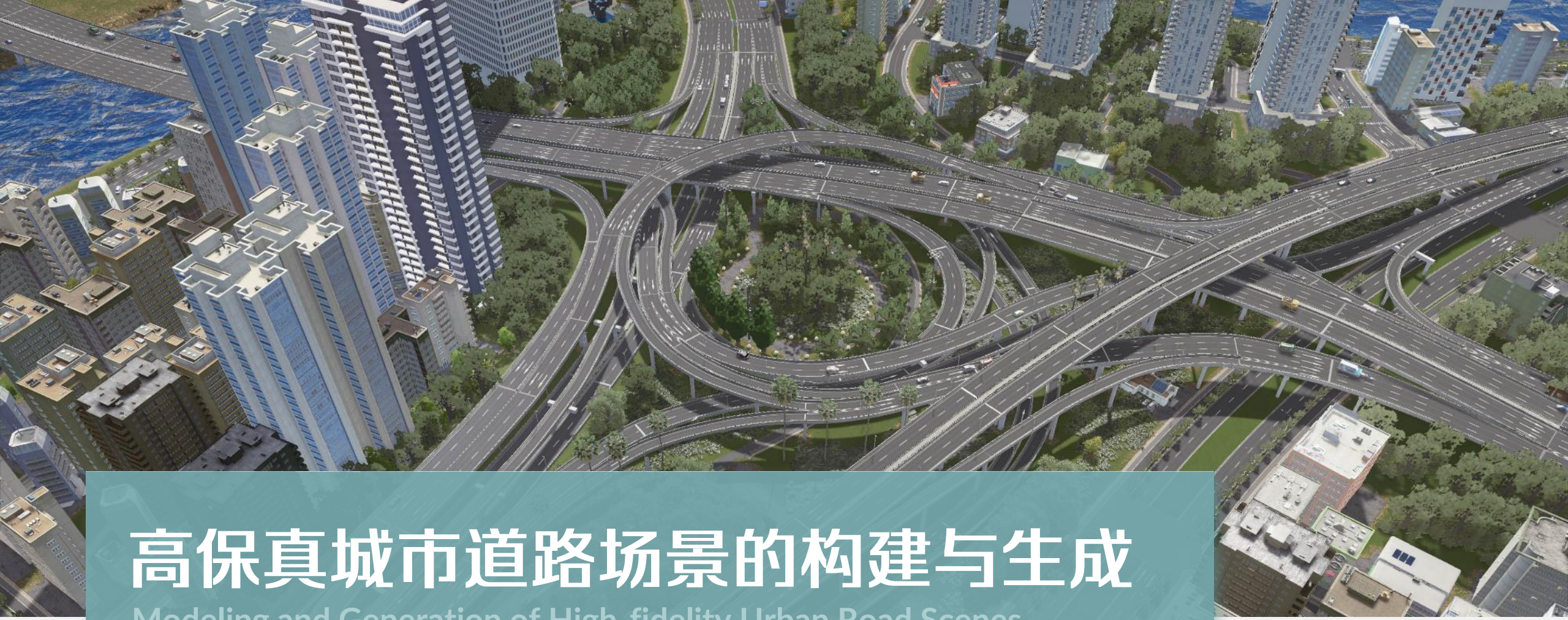


高保真城市道路场景的构建与生成

Modeling and Generation of High-fidelity Urban Road Scenes

Yangzesheng Sun

Email: **sun00032@umn.edu**
Github: **github.com/victoriacity**
Bilibili: **victoriacity74**

目录

Cities: Skylines (2015)

IGN. Cities: Skylines Review, www.youtube.com/watch?v=9xj4ciP0Riw (2015).

Cities: Skylines (2021)

夕林樱花. https://space.bilibili.com/13263262 (2021).

Cities: Skylines by numbers

玩家总数 (PC各平台、主机)

1600 万

Mod 总数

24 万

6.5 亿小时

游戏总时长

4 千亿

存档累计人口

城市模拟游戏的玩法

策略经营

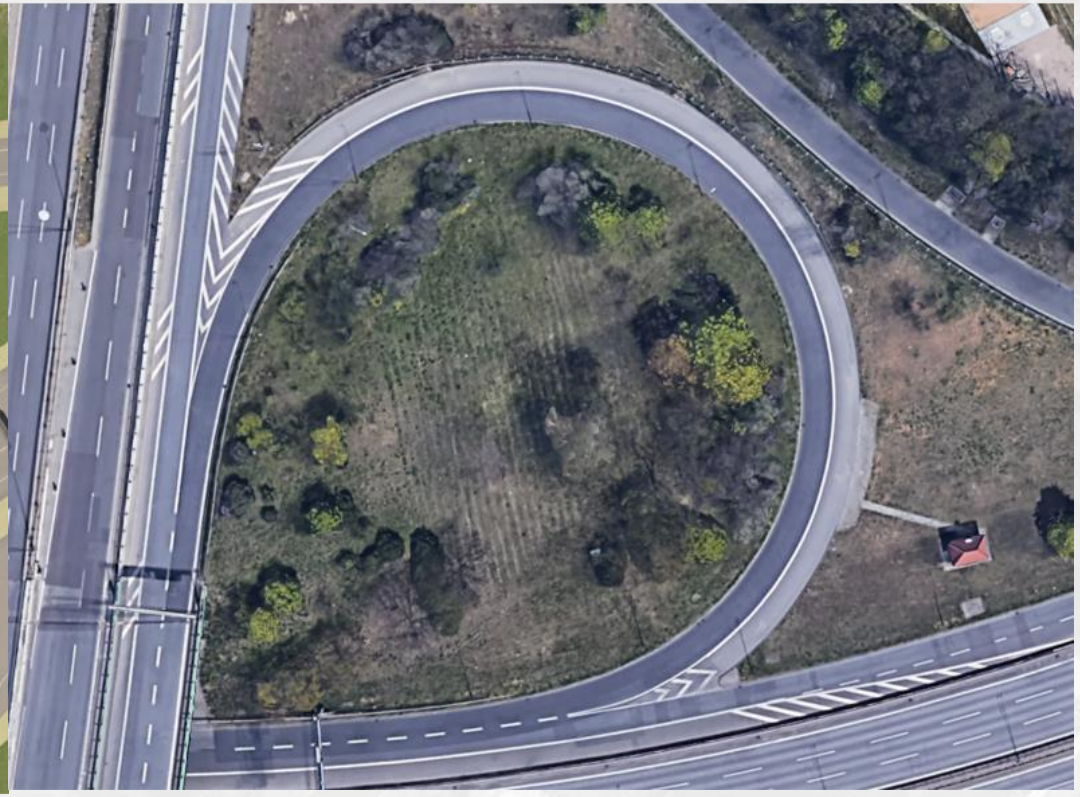
仿真建造

游戏道路工具的局限性

缺乏道路组成和交通模态的多样性

游戏提供的道路难以全面覆盖现实世界中的城市道路组成，如车道数量，停车位，非机动车，有轨电车

游戏道路工具的局限性



缺乏道路衔接处的平滑过渡

不同车道、方向的过渡会使用各种道路标线完成，车道之间会一一对应

目标与解决方案

- **目标**
  - 使用《城市：天际线》建造和现实世界别无二致的城市路网
- **解决办法**
  1. 玩家手工制作道路场景（铺柏油、画标线）
  2. **程序化生成**

目录

其他游戏中的城市道路

Forza Horizon 4

赛车游戏的车道会比现实世界宽

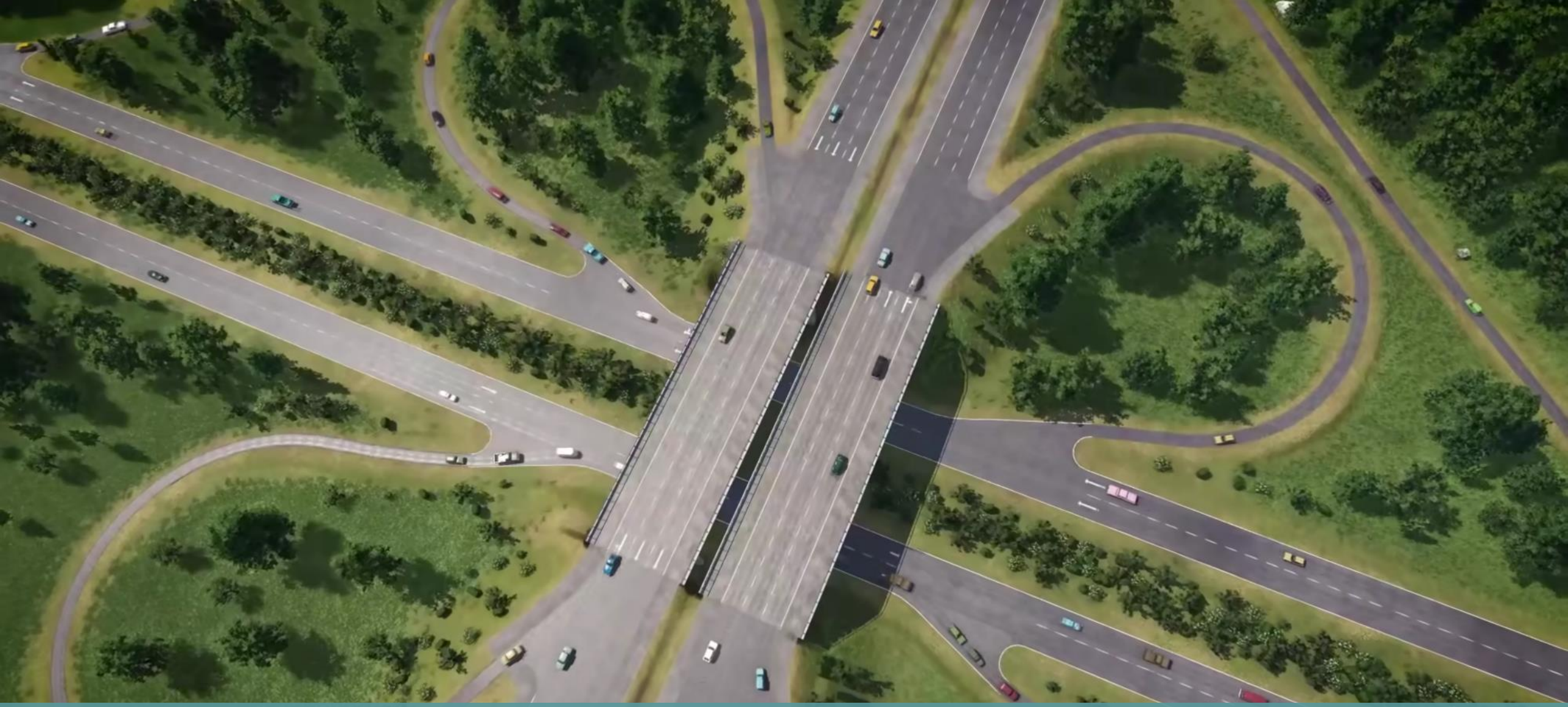
其他游戏中的城市道路

Grid Motorsport (2019)

其他游戏中的城市道路

赛博朋克 2077

拉胯



其他游戏中的城市道路

Transport Fever 2

和原版《城市：天际线》一样没有道路衔接标线

程序化生成路网

- 通过张量场 (Tensor field) 生成路网
- 用电场类比，沿“电场线”和“等势面”确定道路走向
- 路网特征可由用户输入、河岸、地形等确定
- 路网密度使用已知的人口密度数据

Figure 1: *This figure shows the three steps of our pipeline. The input water map is based on a stretch of the Benue River in Nigeria. Left: Starting from topographical water and park maps, the user designs a tensor field. Middle: The tensor field and further editing operations are used to generate a road network. Right: Three-dimensional geometry is created.*

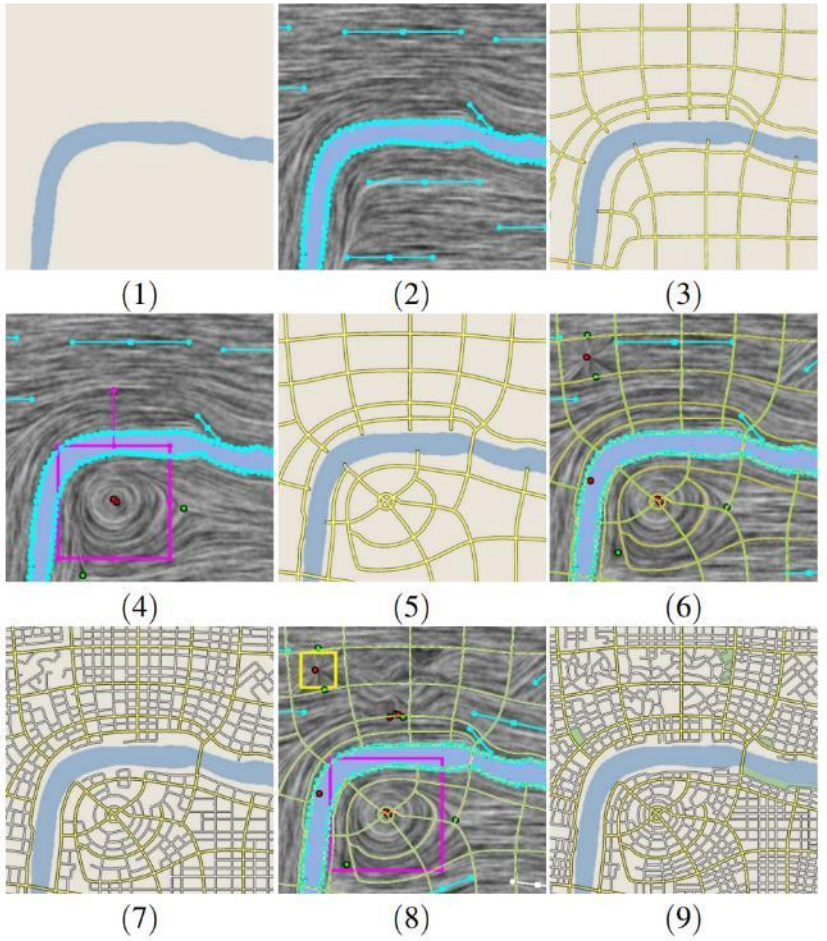
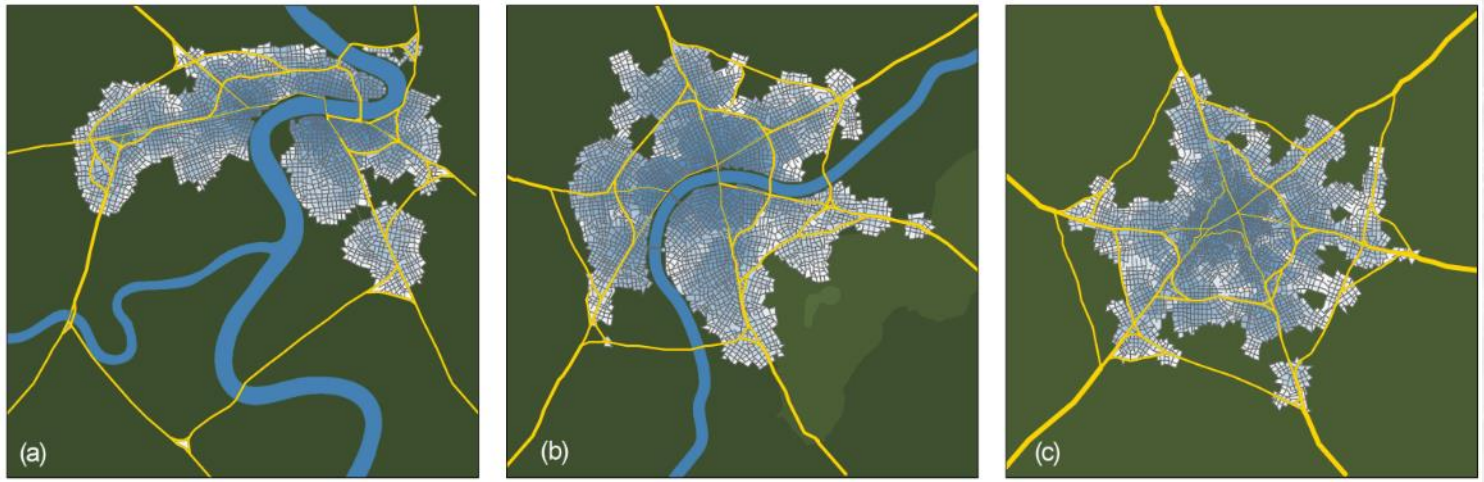


Figure 3: *An example sequence of modeling steps in our system.*

G. Chen *et al.*, Interactive Procedural Street Modeling, *ACM Trans. Graph.* **2008**, 27(3) (SIGGRAPH 2008), 103.

程序化生成路网

- 交通模拟驱动的道路生成
- 用寻路算法确定初始路网和新建道路
- 符合城市模拟的通常做法：
  - 初始路网 – 建城/支路 – 城市发展 – 新建干路



J. Beneš *et al.*, Procedural Modelling of Urban Road Networks, *Computer Graphics Forum* **2014**, 33(6), 132–142.

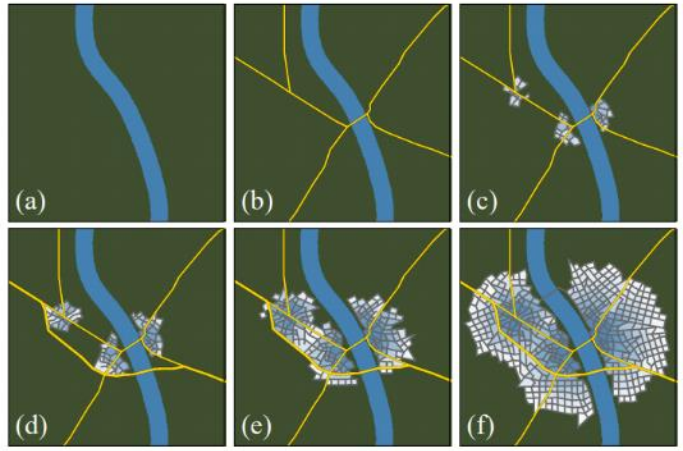


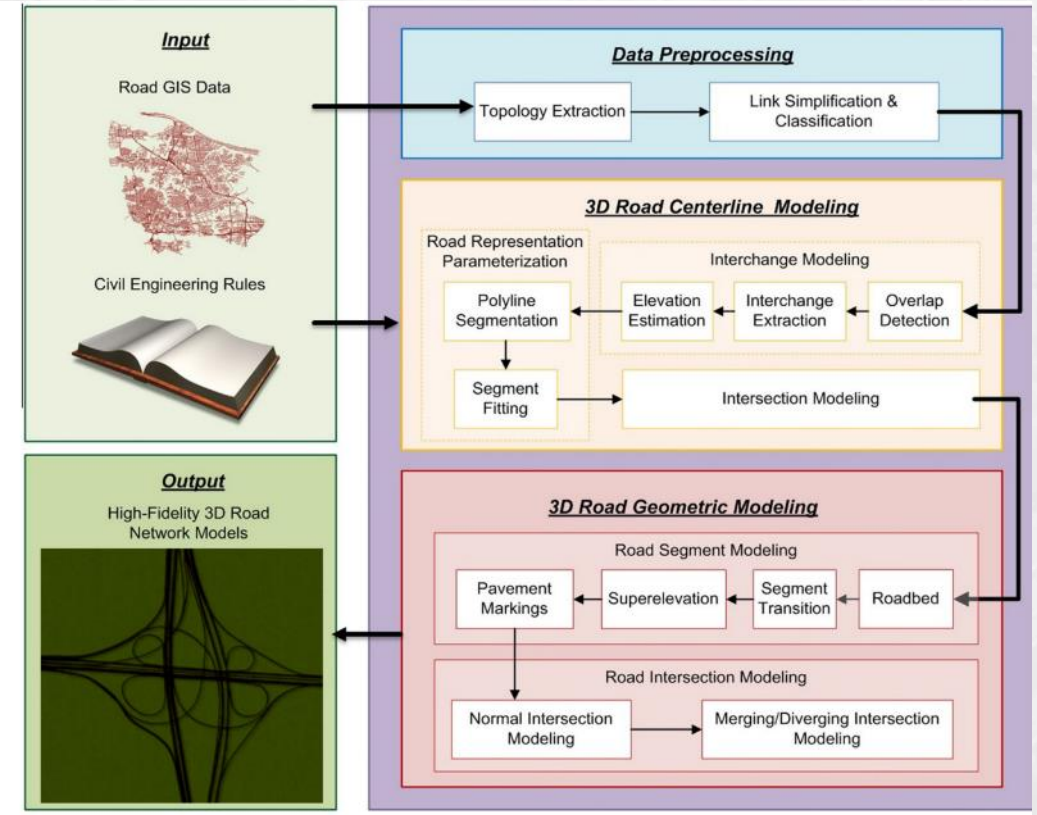
Figure 2: A typical progression of our algorithm. On an empty map (a), an initial road network (b) is grown and nuclei are identified. Initially, the city consists of several smaller settlements (c). As it grows and traffic increases, new major roads (yellow) are built (d) and a conurbation is formed (e). Eventually, even new major roads are absorbed (f).

Algorithm 1: High-level pseudo-code of our algorithm.

```
INITIALIZATION();                    // see Section 3.2
while t < t_max do
    GROWMINORROADS();                // see Section 3.3
    TRAFFICSIMULATION();             // see Section 3.4.2
    GROWMAJORROADS();                // see Section 3.4.2
    INCREASE(t);
end
```

程序化生成路网

- 根据 GIS 数据计算道路线形和三维模型



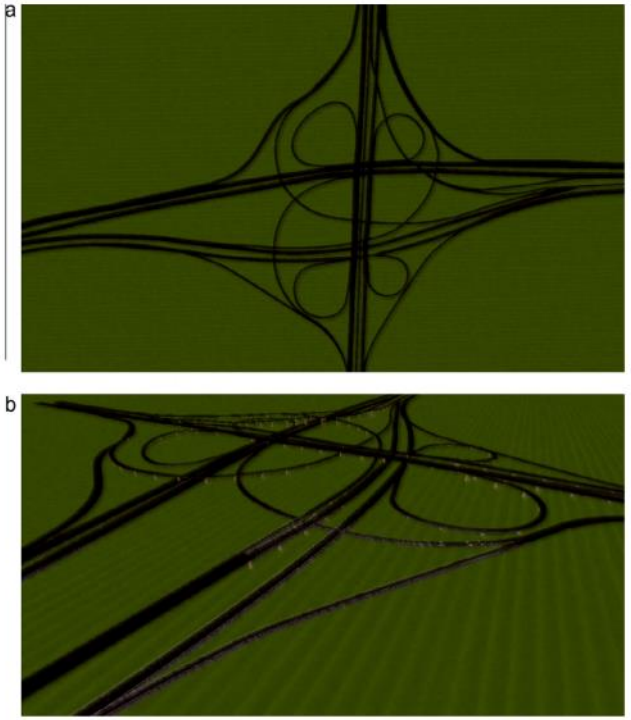


Fig. 16. Top view of the I-64 I-264 interchange in Norfolk, Virginia, USA (a) and an aerial view of the same interchange including support structures.

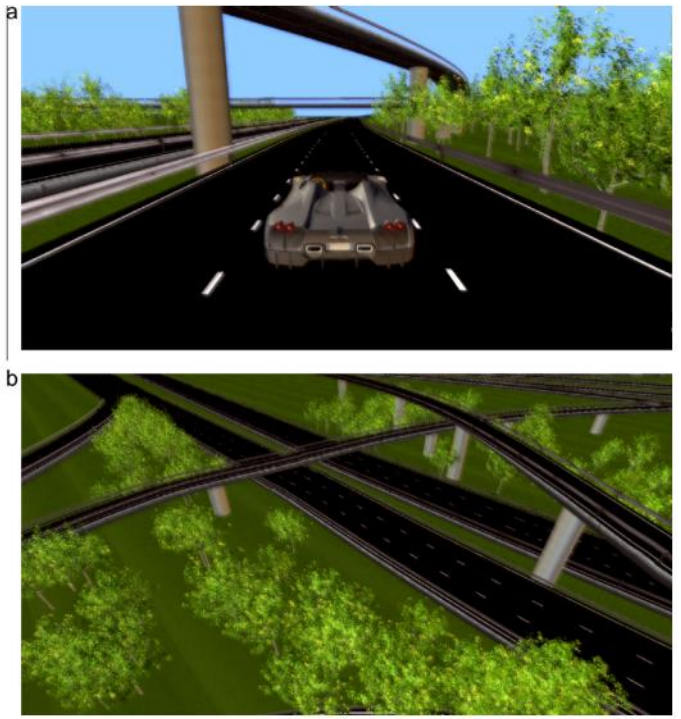


Fig. 17. Detailed views of the I-64 I-264 interchange featuring additional aesthetic enhancements.

J. Wang *et al.*, Automatic high-fidelity 3D road network modeling based on 2D GIS data, *Advances in Engineering Software.* **2014**, 76, 86–98.

模块化道路

旧版 CSUR (2018)

- **“烘焙”道路过渡**，把各种道路的衔接类型做成静态的资产
- **引入车道编号的概念**，为了满足游戏机制，不同位置的车道对应不同的道路资产
- **模块化道路**，把手画标线的“针线活”变成拼接模块的“搭积木”
- 人工制作数十个道路资产，美术工作量巨大

目录

CSUR 项目概要

- Cities: Skylines Urban Road, 简称 **CSUR**，是《城市：天际线》的一套资产 mod 合集。CSUR 为城市模拟游戏带来了**前所未有的拟真程度**。
- CSUR 是一套**离线程序化生成**系统，基于一套顶层的道路设计 API 和 Blender 图形后端，以及 Unity 运行时支持。

- **总计**用户数 (Steam工坊)：3.7 万
- 活跃用户数：**少** (**~1%** 全游戏月活)

CITIES: SKYLINES URBAN ROAD

视觉形象设计
Visual Identity Design

高速公路标识

+

复杂道路网络

+

模块化组件

=

CSUR 主图标
「彩虹岔路」

CSUR 绿
RGB: 44 181 29
CMYK: 77 0 100 0
HEX: #2CB51D

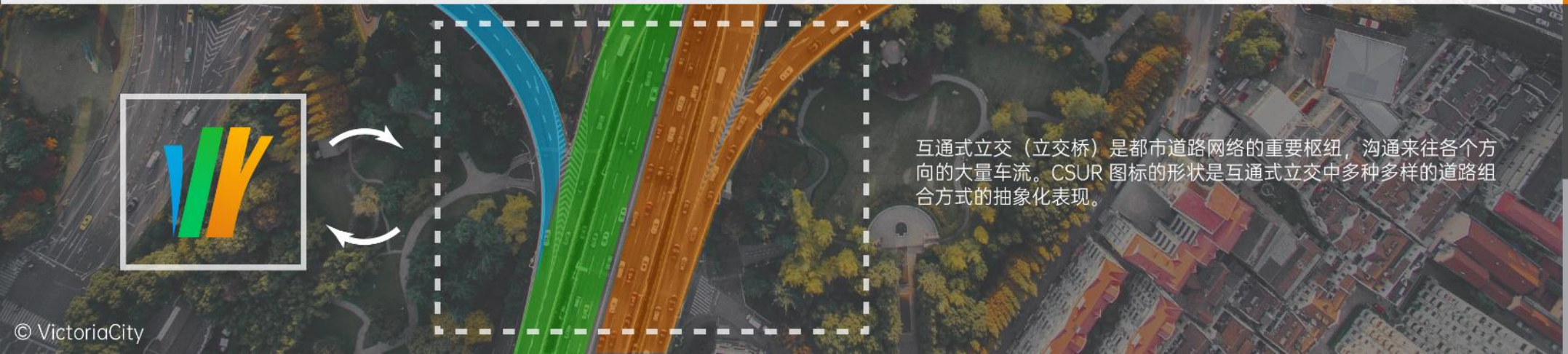
CSUR 紫
RGB: 221 74 221
CMYK: 29 77 0 0
HEX: #DD4ADD

CSUR 青
RGB: 5 166 221
CMYK: 74 17 1 0
HEX: #05A6DD

CSUR 橙
RGB: 232 128 15
CMYK: 6 59 100 0
HEX: #E8800F

辅助色 深灰
RGB: 77 77 77
CMYK: 0 0 0 85
HEX: #4D4D4D

辅助色 浅灰
RGB: 168 168 168
CMYK: 0 0 0 42
HEX: #A8A8A8

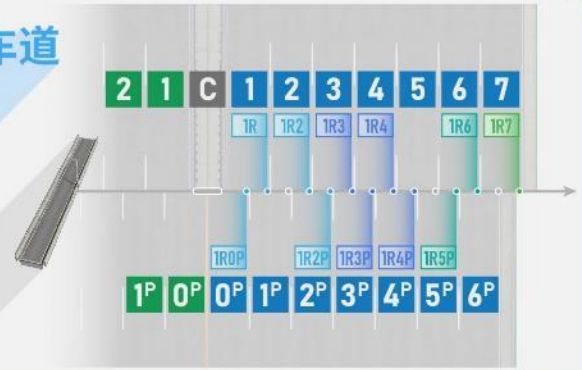


互通式立交（立交桥）是都市道路网络的重要枢纽，沟通来往各个方向的大量车流。CSUR 图标的形状是互通式立交中多种多样的道路组合方式的抽象化表现。

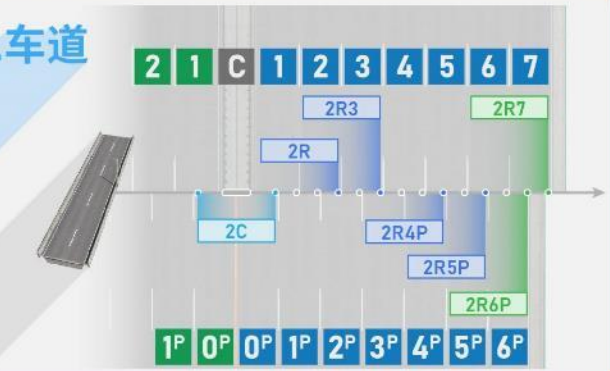
CSUR
CITIES: SKYLINES URBAN ROAD

道路模块速查表

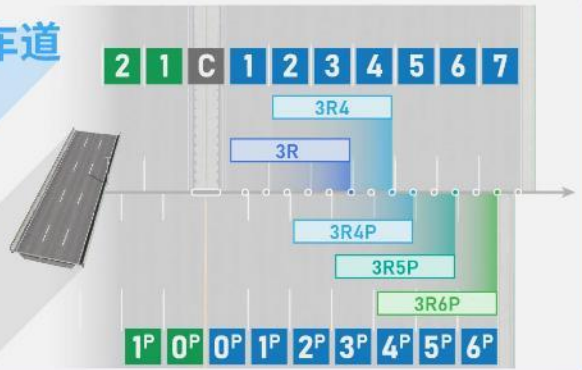
一车道



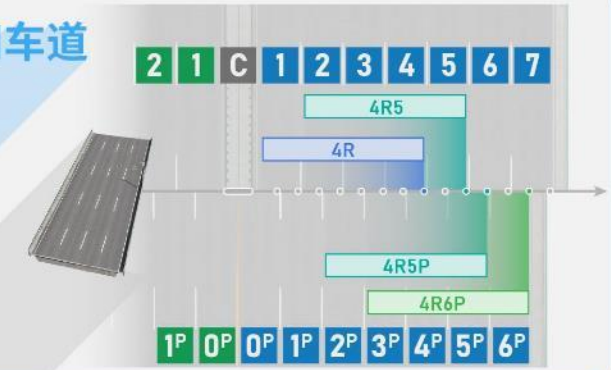
二车道



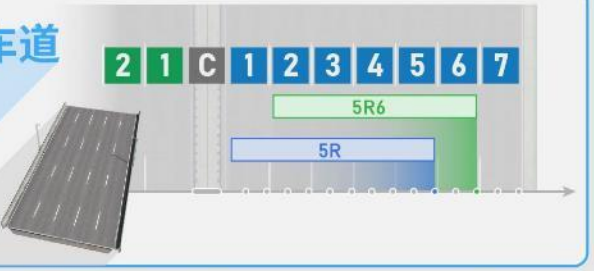
三车道



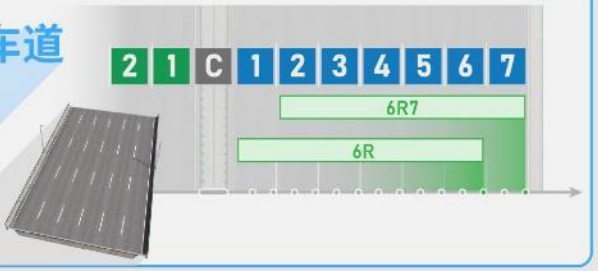
四车道



五车道



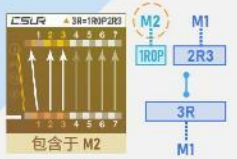
六车道



道路模块分类

道路包内容

在左栏中找到模块两端车道所在的格子，模块所属的道路包由格子中图例颜色的最大号道路包决定。



图例

其他道路包

R1 包含到 M4 (5P位置) 范围无分隔带或窄分隔带的双向道路

R2 包含到 M4 (5P位置) 范围宽分隔带的双向道路

R3 包含到 M5 (7位置) 范围的双向道路

R4 包含不对称双向道路

M3 包含无分隔带的道路模块

B 包含中央为2车道五分隔带的快速公交道路

选择无分隔带道路

[R1] [R2][R3]
提供 2, 4, 6, 8 车道

选择居中单行道

[R1][R2]
提供 1, 2, 3, 4 车道

请务必订阅并打开 CSUR Loader 和 CSUR Toolbox!

大型立交桥

快速公交 (BRT)

和现实路网精确到车道吻合

System overview

CITIES: SKYLINES UR BAN ROAD

dependent → dependency
input ⇢ output

Cairo required

cairo

Graphics

3 UI Thumbnail

User input or script
`./make 3R`

Builder

Core

Prefabs

Road configuration data

```
u = [BARRIER, LANE, …]
x = [0.9375, 1.875, …]
```

3
2
1

CITIES SKYLINES

unity

Blender required

Modeling

1 Meshes

2 Prefab data

`<XML></XML>`

C#/ Unity Plugins*

Offline generation

Runtime

*https://github.com/citiesskylines-csur/{RoadImporter, CSURLoader, CSURToolbox}

目录

游戏场景中的路网

有向图：

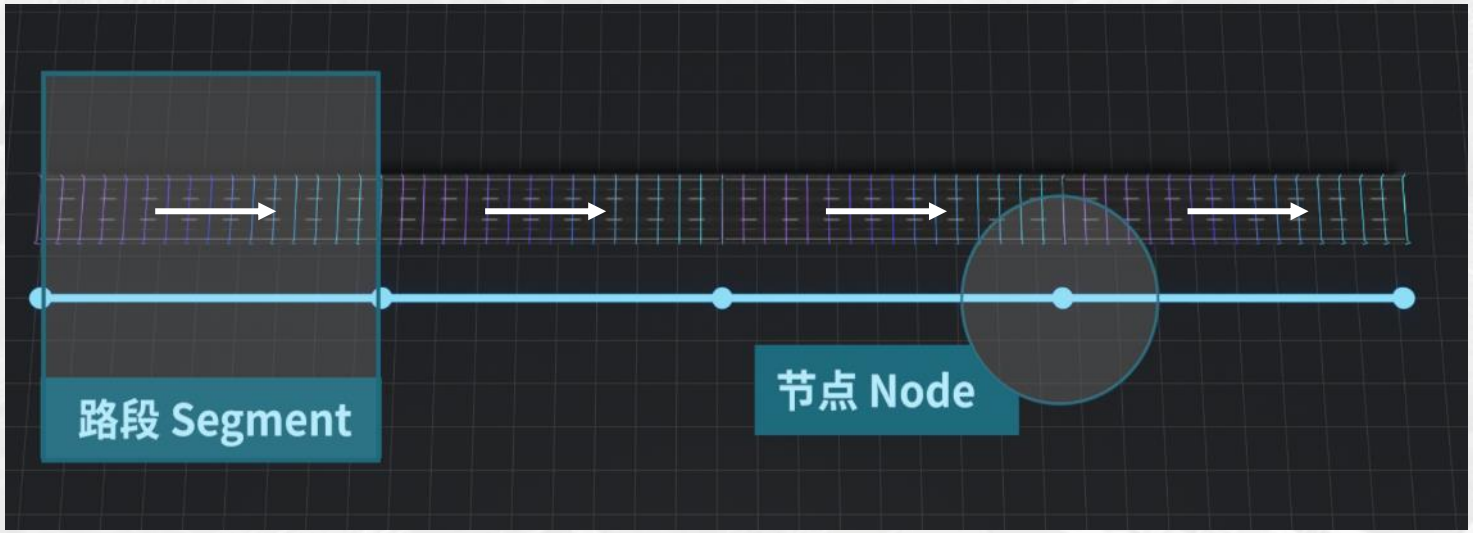
- Vertex – 节点
- Edge – 路段

节点属性

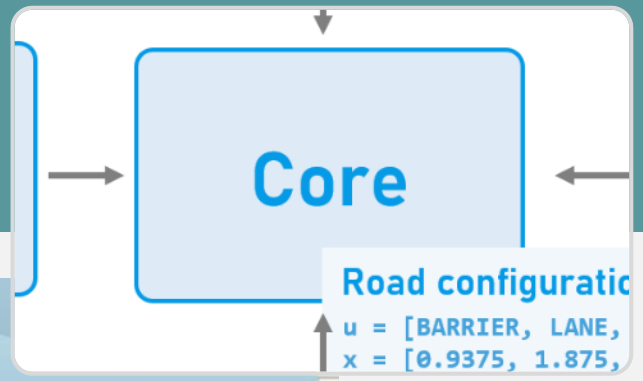
- 位置
- 是否有路口
- …

路段属性

- 起始方向
- 中止方向
- **道路组成**
- …

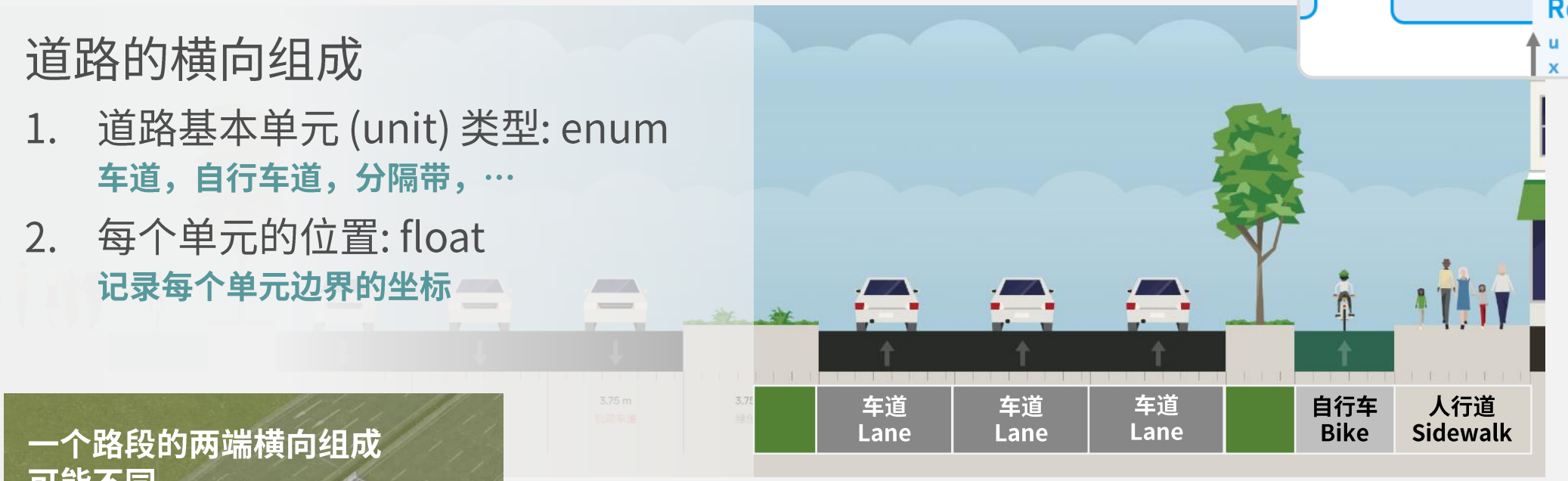


道路组成的表示

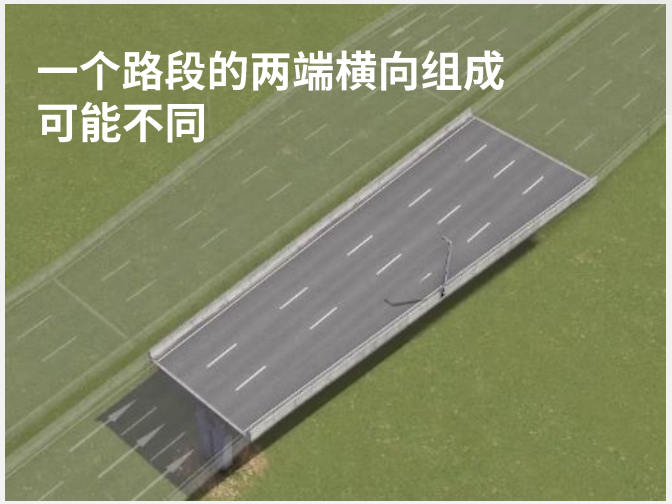


道路的横向组成

1. 道路基本单元 (unit) 类型: enum
   车道，自行车道，分隔带，…
2. 每个单元的位置: float
   记录每个单元边界的坐标




```
self.units = [MEDIAN, LANE, LANE, LANE, MEDIAN, BIKE, SIDEWALK]
self.x = [0, 0.5, 1.5, 2.5, 3.5, 4, 4.75, 5.75] * LANE_WIDTH
```



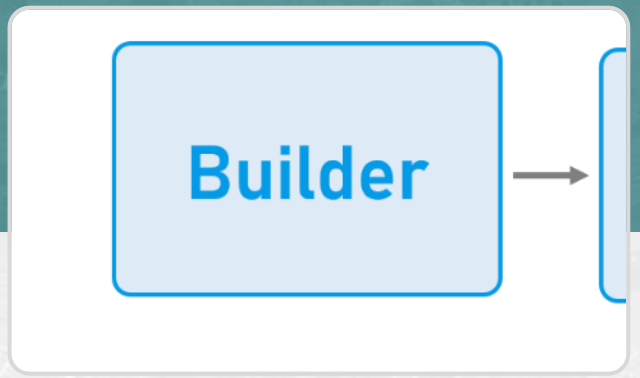

```
self.units_start = …
self.units_end = …
```

每个单元对应一个submesh生成道路模型

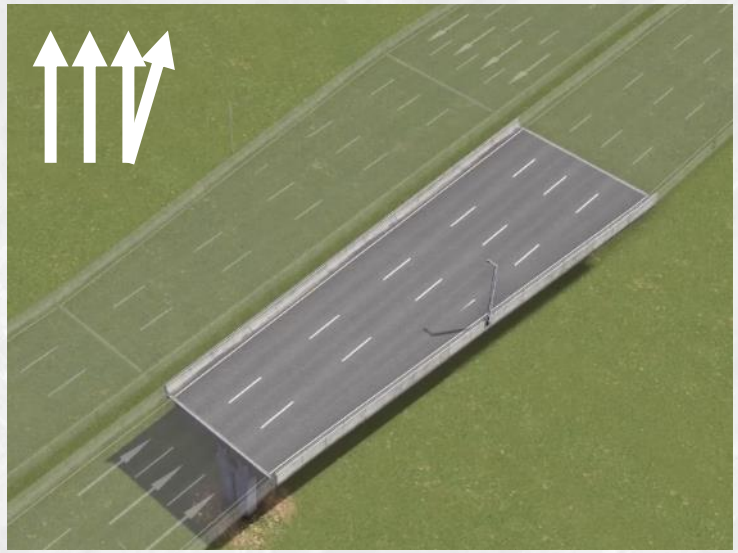
要求在道路过渡时

`len(units_start) == len(units_end)`

车道衔接推断



- 假设**除行车道之外的组成单元不变**（护栏、人行道等）
- 引入空单元类型 **EMPTY = 0**，宽度为0
  - 判定拼接 submesh 的模型类型使用 `units_start[i] or units_end[i]`
  - EMPTY 不影响车道数量的计算



```
units_start = [BARRIER, LANE, LANE, LANE, EMPTY, BARRIER]
units_end =   [BARRIER, LANE, LANE, LANE, LANE,  BARRIER]

x_start =   [0.25,   0.5,  1.5,  2.5,  3.5,   3.5,   3.75]
x_end =     [0.25,   0.5,  1.5,  2.5,  3.5,   4.5,   4.75]
```

功能：**3车道在最右侧增加1条车道**
用不同的EMPTY位置可以改变增加车道的方向

4
3

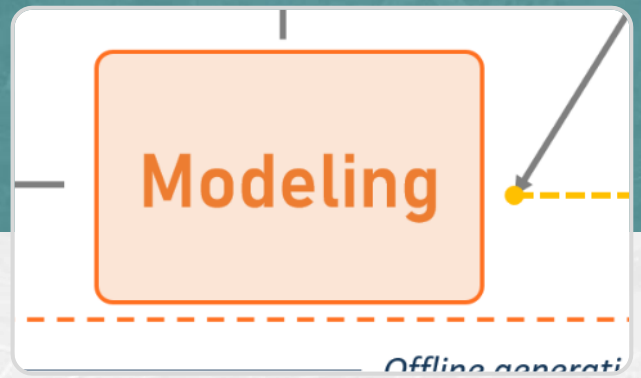
车道衔接推断

- 道路的分流与合流：使用**导流线**标记岔路口
- 导流线纹理：V形斑马线条纹

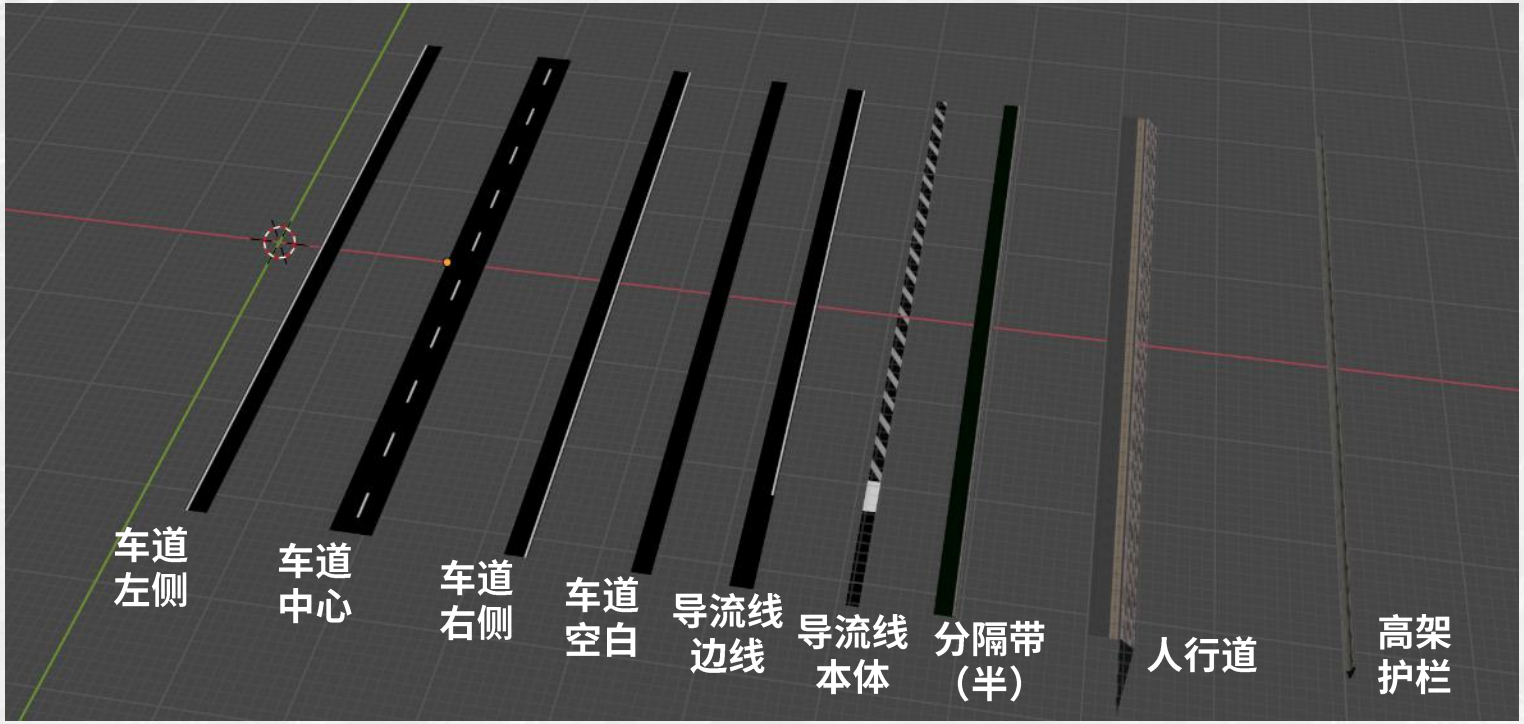
```
units_start = [BARRIER, LANE, LANE, EMPTY,   LANE, LANE, BARRIER]
units_end =   [BARRIER, LANE, LANE, CHANNEL, LANE, LANE, BARRIER]

x_start =   [0.25,   0.5,  1.5,   2.5,   2.5,   3.5,  4.5,   4.75]
x_end =     [0.25,   0.5,  1.5,   2.5,   3.0,   4.0,  5.0,   5.25]
```

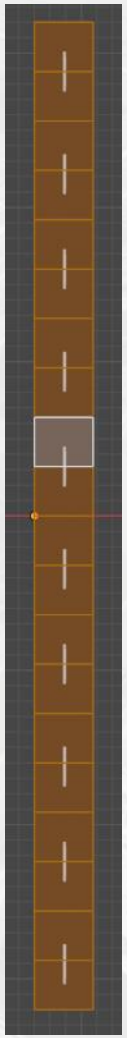
不同的分岔方式：

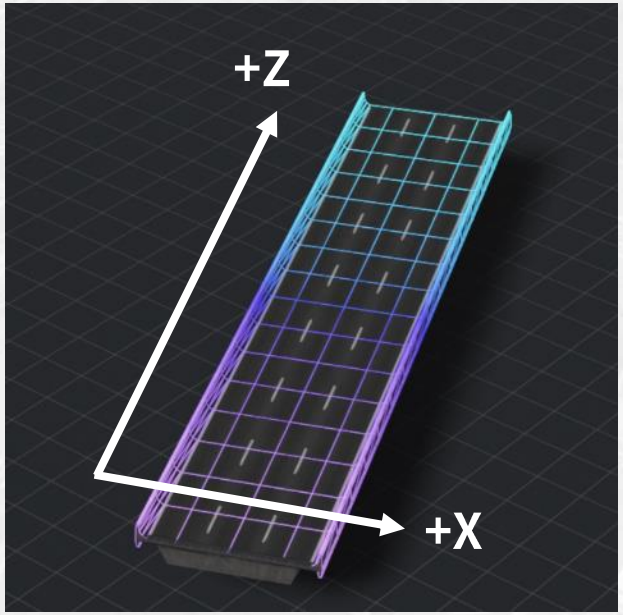
模型几何处理



- 道路基准模型沿 +Z 方向细分，**每个分段 (subdivision) 平行于 X 轴**
- 用已分段的各个单元 submesh 拼接成道路 mesh

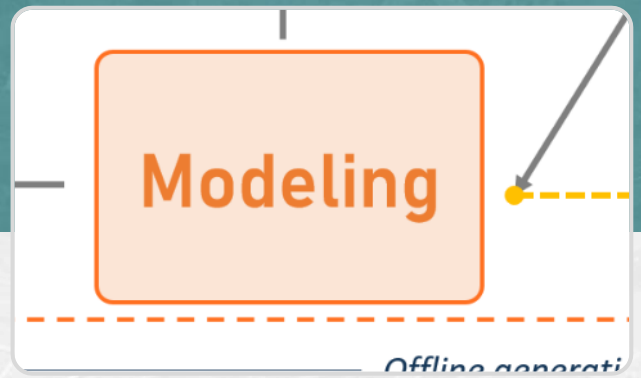




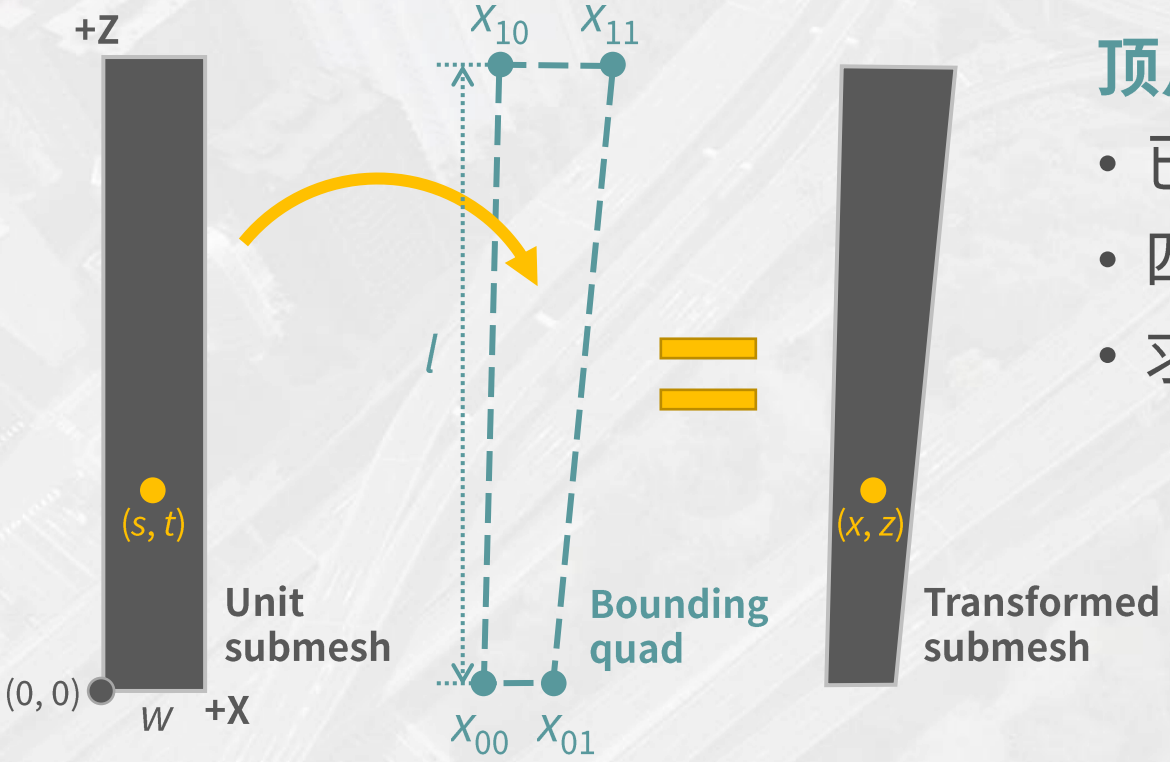


Unity/OpenGL **左手系：**
+X 向右, +Y 向上, +Z 向前

模型几何处理



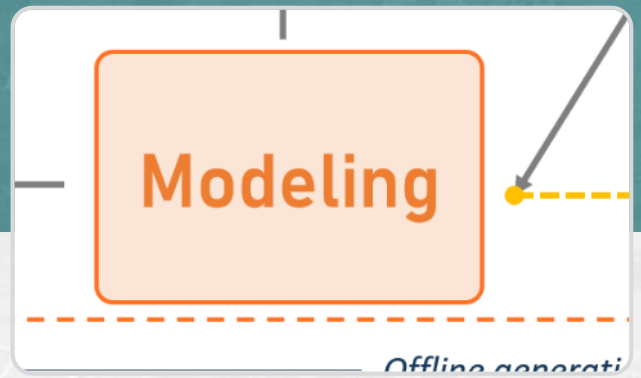
- Submesh 的**宽度伸缩**
- 把单元填充满四边形边界 ($x_{00} \le x_{01}, x_{10} \le x_{11}$)



顶点坐标变换

- 已知原submesh某个顶点的分数坐标 (s, t)
- 四边形尺寸 (x_{00}, x_{01}, x_{10}, x_{11}, l)
- 求变换之后顶点的坐标 (x, z)
  - y 坐标相对原submesh不变

模型几何处理

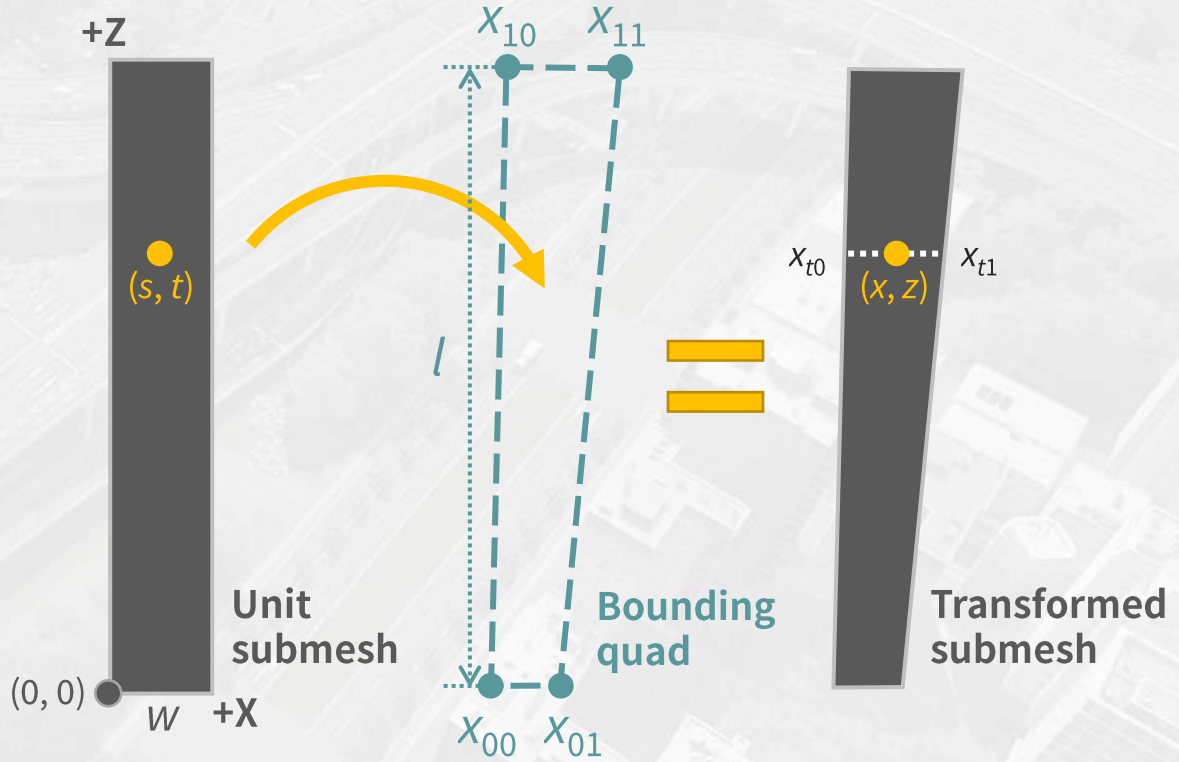


- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出**

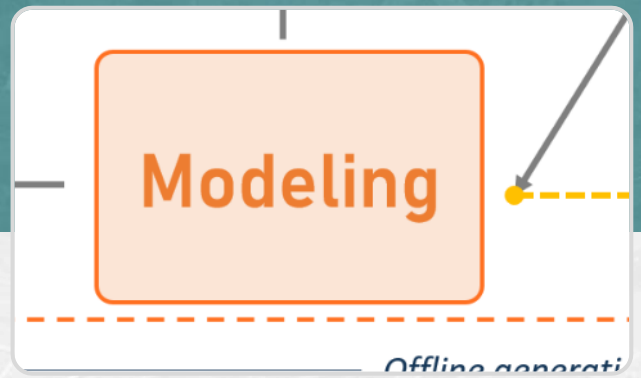
1. 中心拆分

- 在中心线和左(-X)侧的顶点保持到左边界距离不变，反之亦然
- $$x = \begin{cases} x_{t0} + sw_0, & 0 \le s \le 0.5 \\ x_{t1} - (1-s)w_0, & 0.5 < s \le 1 \end{cases}$$

要求模型的原始宽度 w 不大于变换后的最小宽度！



模型几何处理

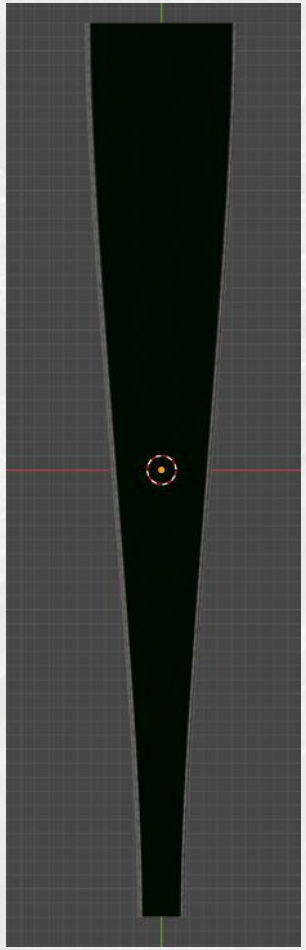


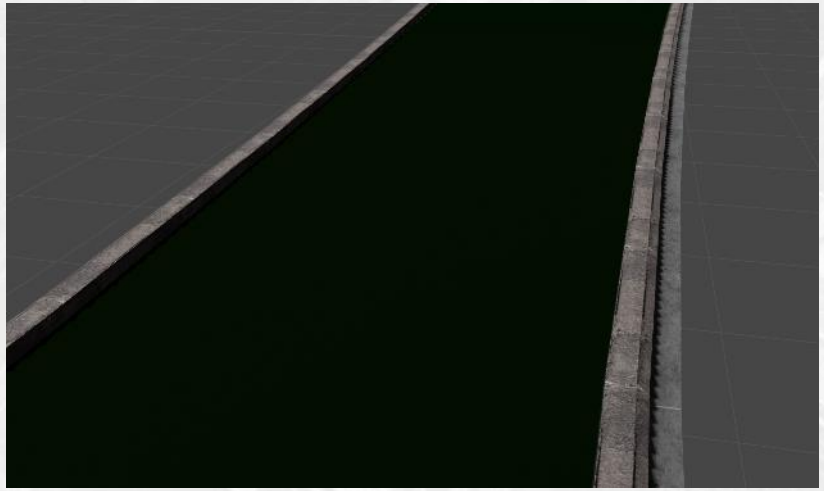
- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出**

1. 中心拆分

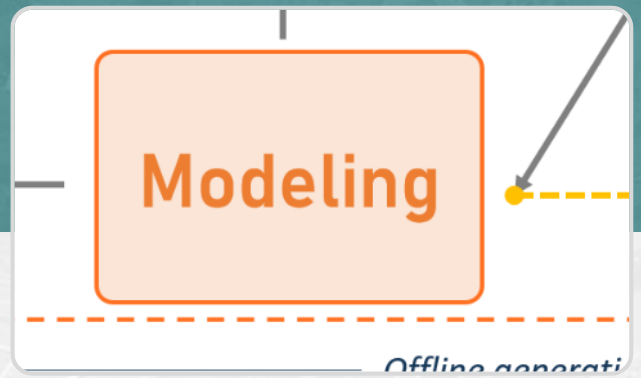
- 在中心线和左(-X)侧的顶点保持到左边界距离不变，反之亦然
- $$x = \begin{cases} x_{s0} + sw_0, & 0 \le s \le 0.5 \\ x_{s1} - (1-s)w_0, & 0.5 < s \le 1 \end{cases}$$

用于中央分隔带、护栏，隧道等的变换
保持路缘石（马路牙子）的宽度和隧道墙厚度等不变





模型几何处理

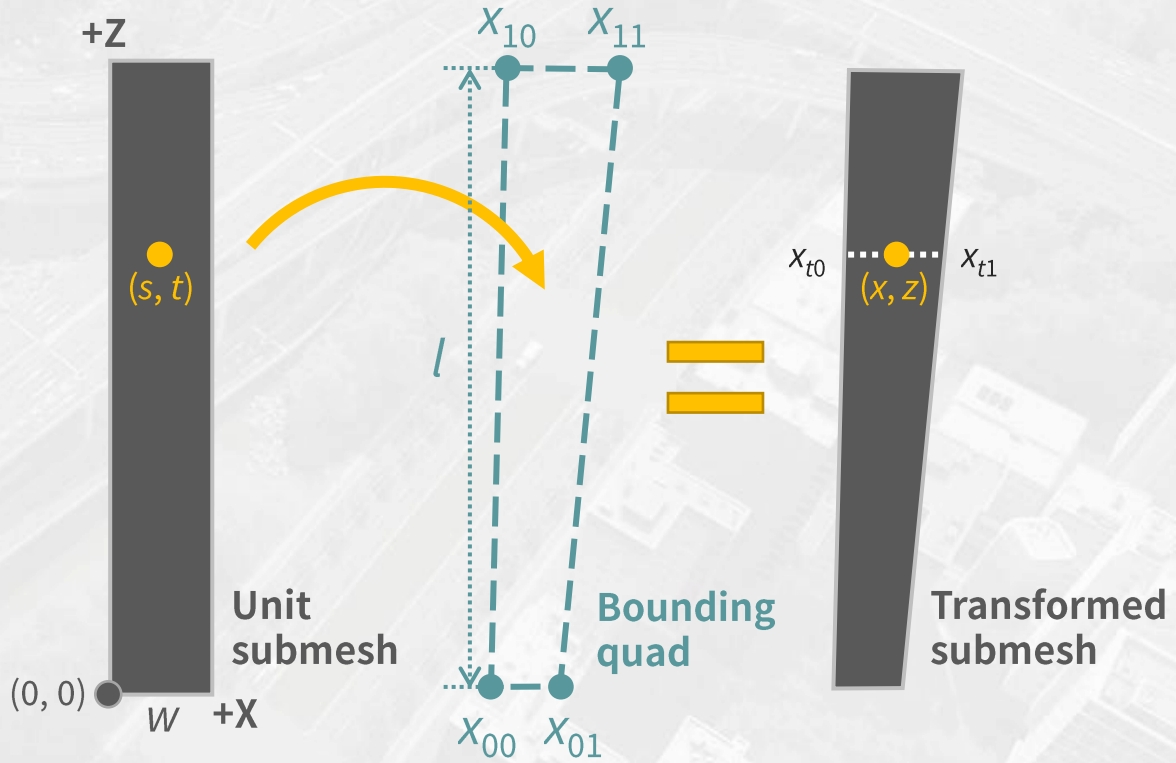


- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出**

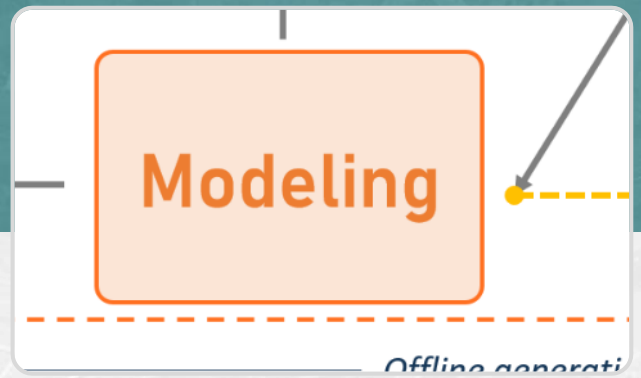
2. 等比缩放

- 线性插值
- $x = (1 - s)x_{s0} + sx_{s1}$

对模型尺寸没有限制



模型几何处理

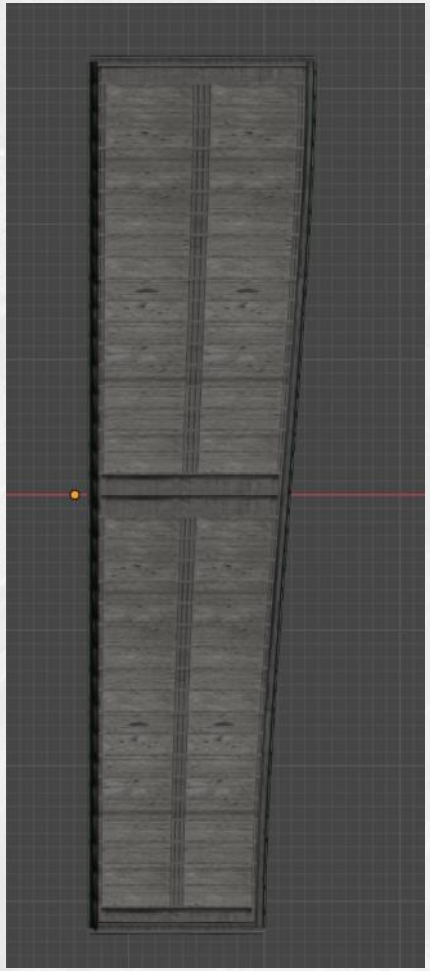


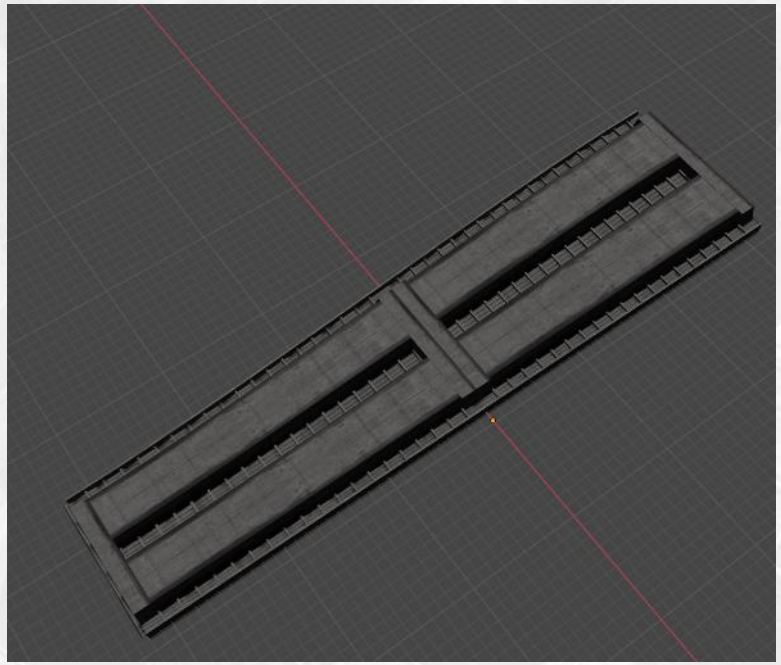
- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出**

2. 等比缩放

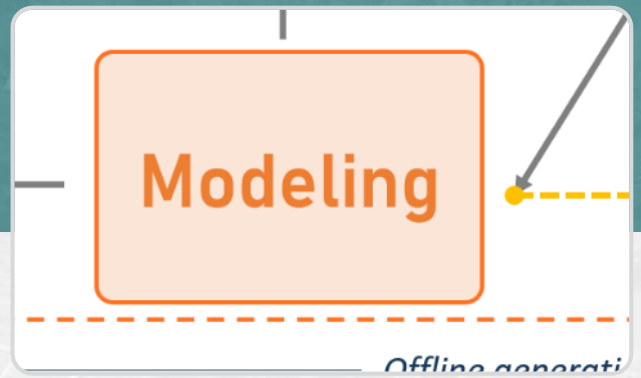
- 线性插值
- $x = (1 - s)x_{s0} + sx_{s1}$

用于高架桥纵梁等可以自由缩放的部件变换



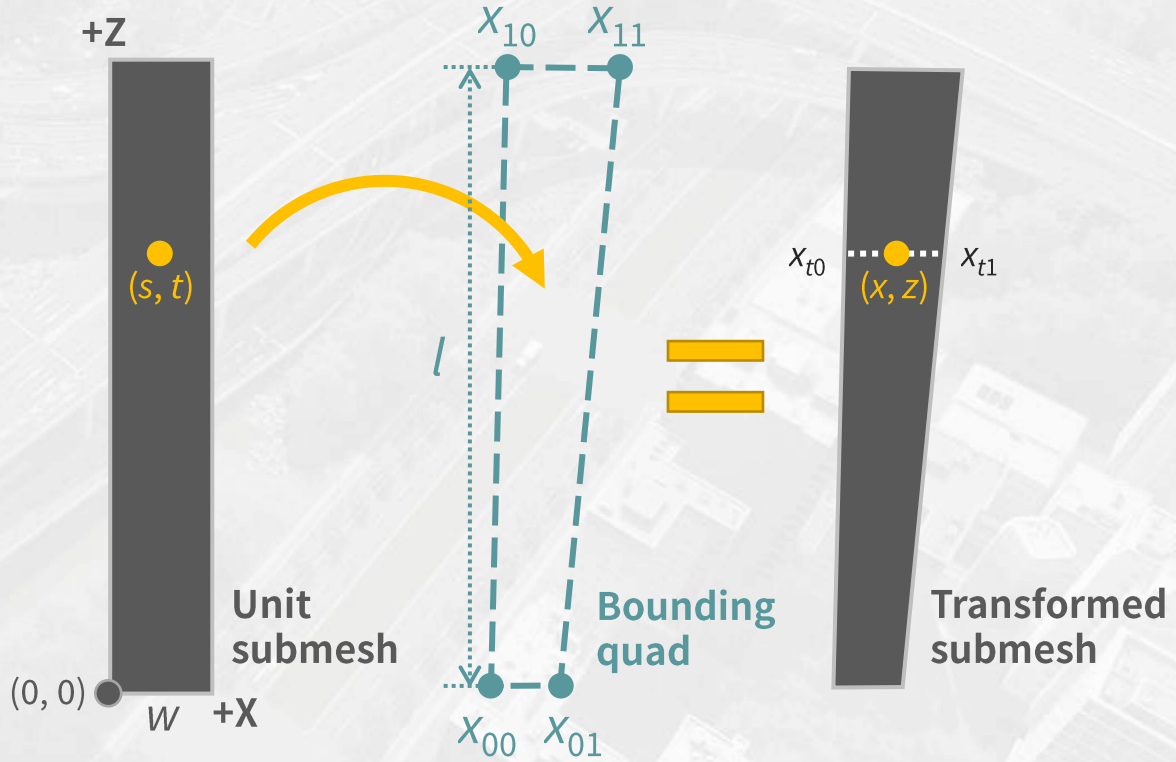


模型几何处理

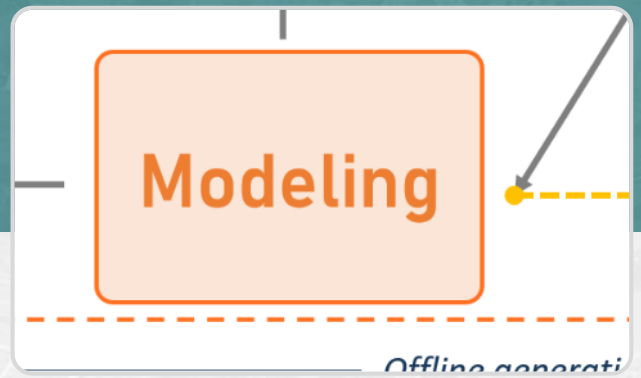


- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出 (怎么求?)**
- **直觉：线性插值**
  - $x_{t0} = (1-t)x_{00} + tx_{10}$
  - $x_{t1} = (1-t)x_{01} + tx_{11}$

游戏用Bezier曲线生成弯曲道路
和线性插值的叠加效果不好

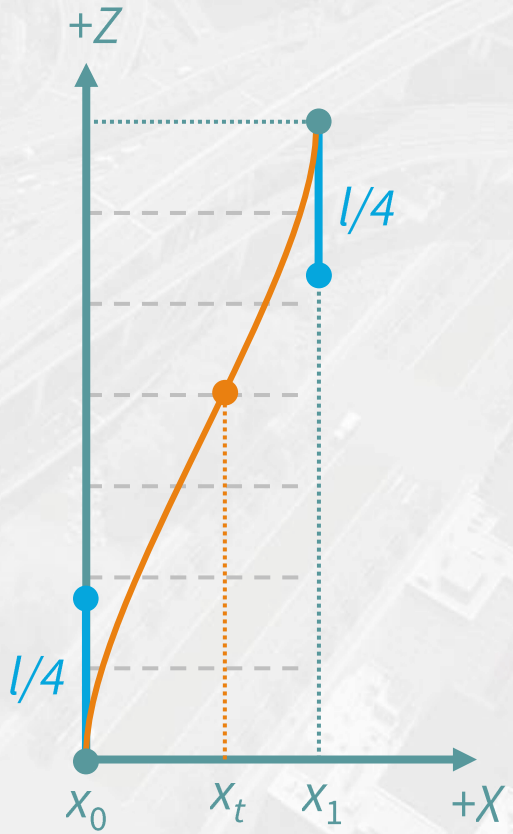


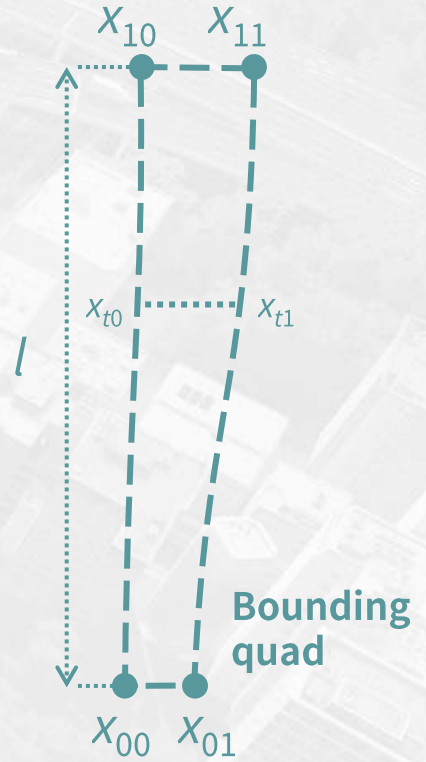
模型几何处理



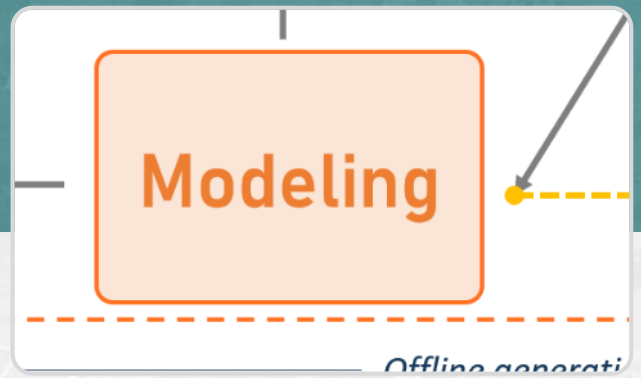
- z 方向
  - $z = t/l$
- x 方向
  - 假设 z 处的X轴两端点为 x_{t0} 和 x_{t1}
  - x_{t0} 和 x_{t1} 可以**通过插值求出 (怎么求?)**
- **解决方案：Bezier插值**
  - “Quad” 的 Z 方向两边使用三次Bezier曲线
  - 两个中间控制点在 1/4 *l* 和 3/4 *l* 处
  - $x_{s0} = \text{sample\_bezier\_z}(x_{00}, x_{10}, t)$
  - $x_{s1} = \text{sample\_bezier\_z}(x_{10}, x_{11}, t)$

需使用三次方程求根公式 (过程略)



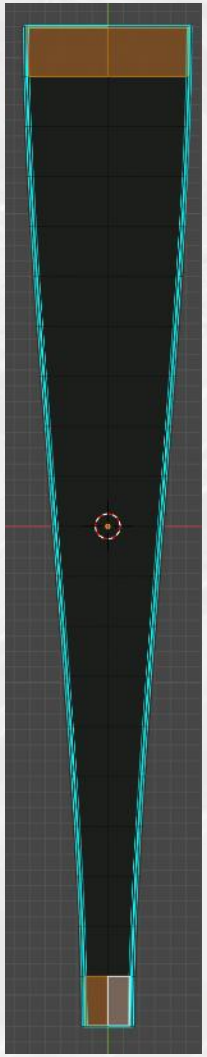


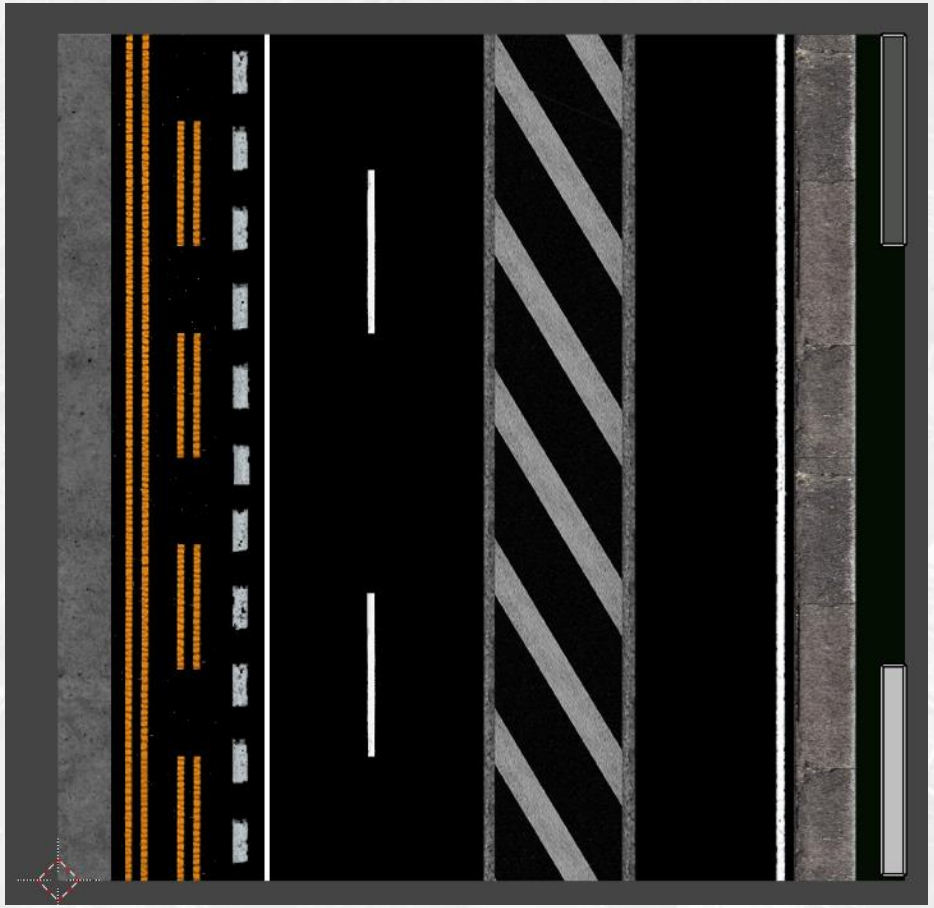
模型几何处理



材质坐标变换

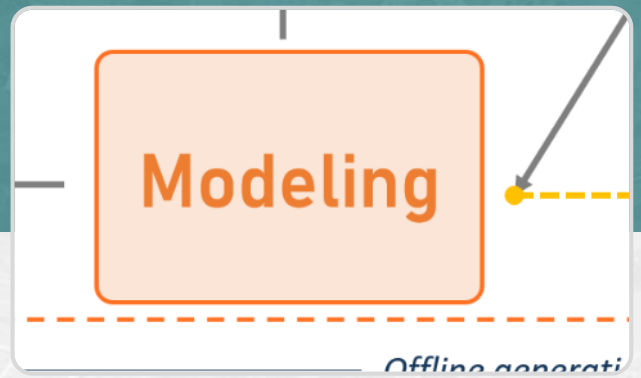
1. UV 坐标不变（普通伸缩）





中央分隔带宽度不同的两组面占据 UV 坐标的宽度相同
(游戏对草地使用特殊shader)

模型几何处理

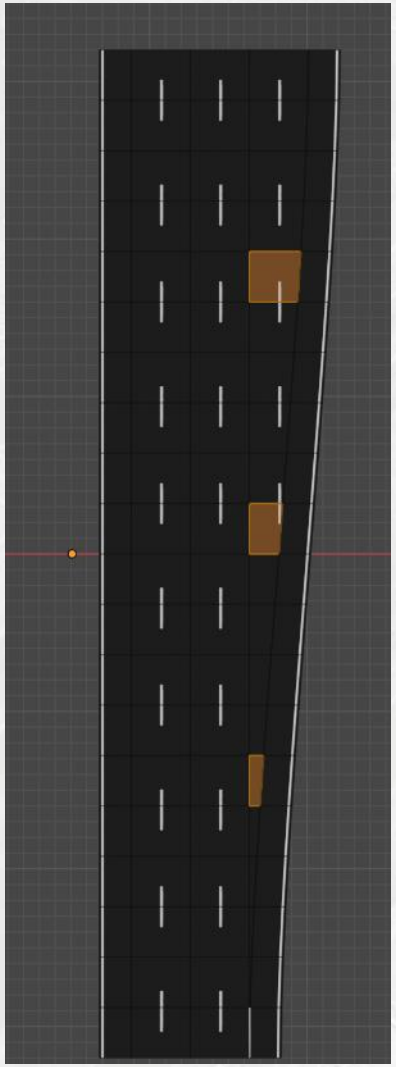


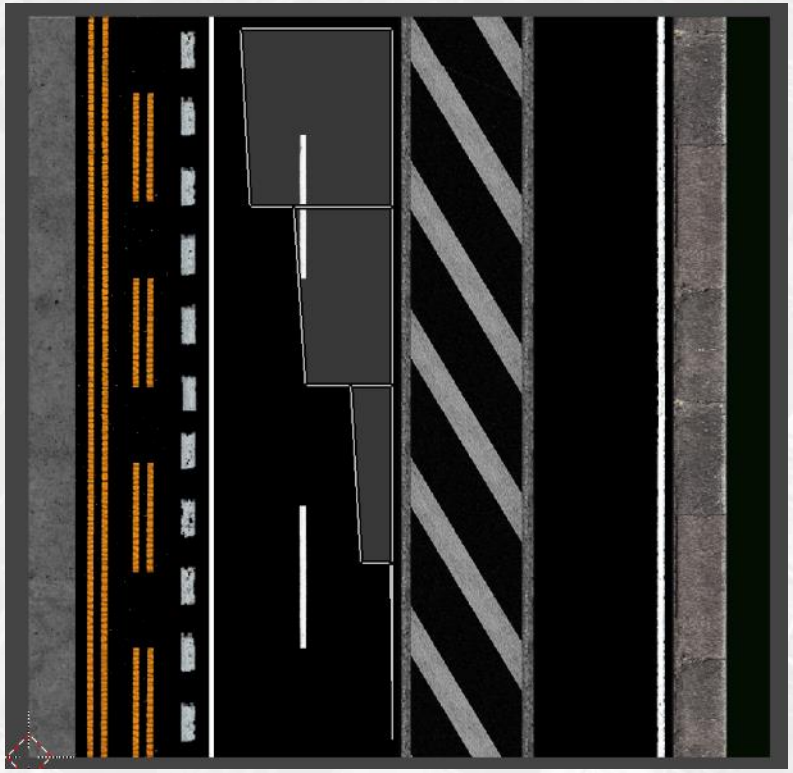
材质坐标变换

1. UV 坐标不变（普通伸缩）
2. **UV 坐标和顶点相同变换 (Vertex slide)**

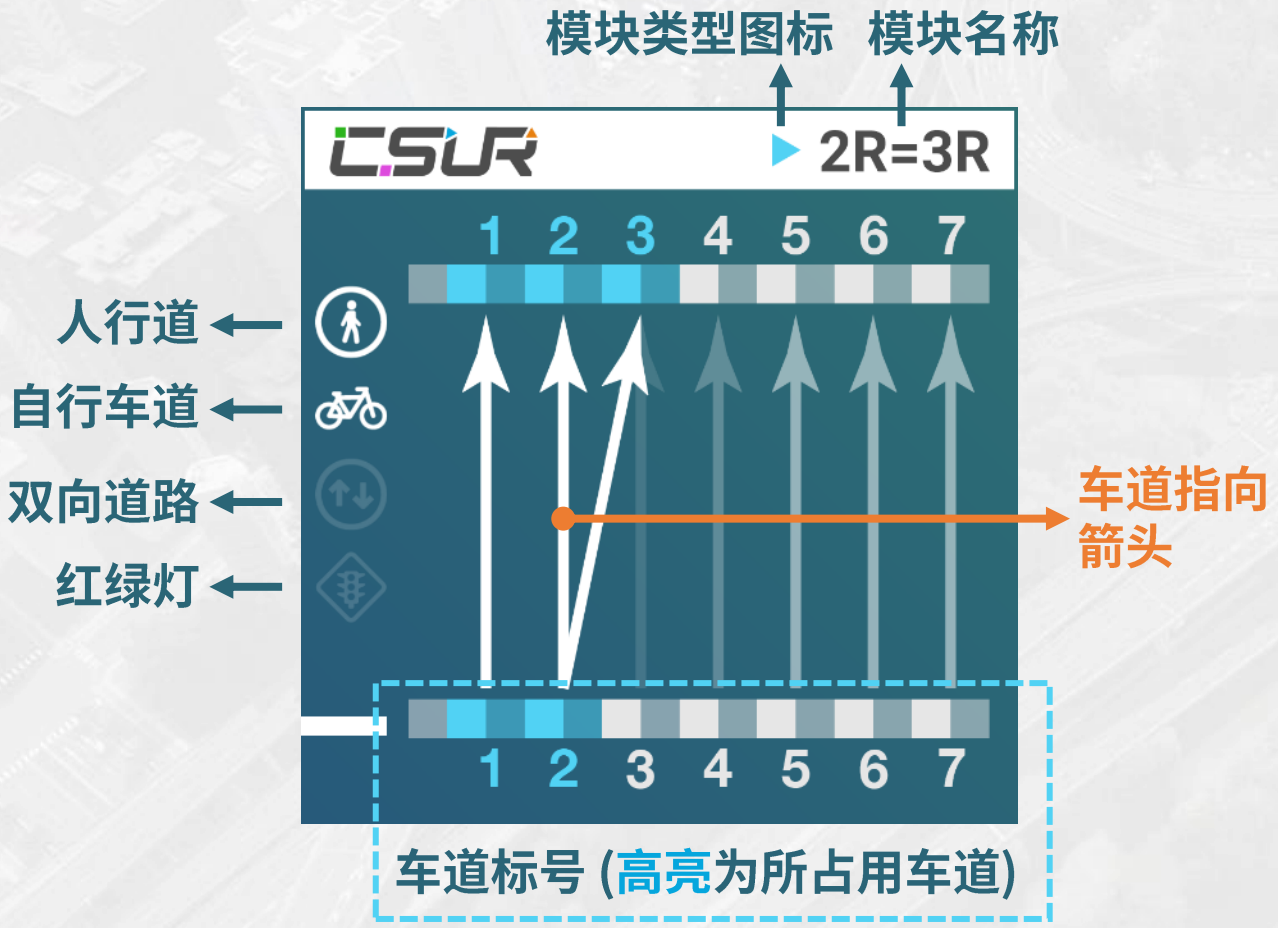
裁切匝道、导流线等单元

Blender 的 Vertex slide功能有限，需重新实现



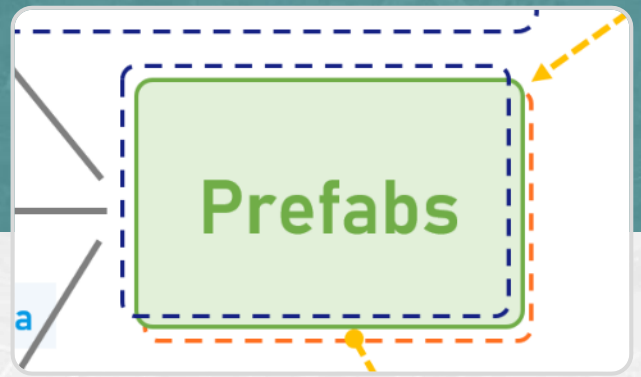


资产功能可视化



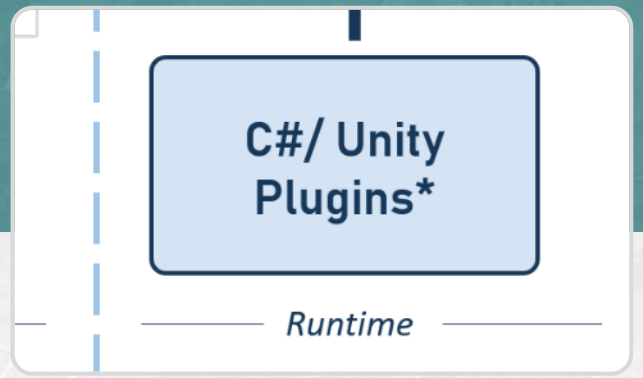
- **2D图形库**
  - Cairo 后端 + Python bindings
  - 锚点 (Anchor) 位置
  - 画布 (Canvas)
  - 纯色与渐变填充
  - 添加图片
  - 添加多边形
  - 添加线段和箭头
  - 添加文字
- Design choices
  - Cairo / Matplotlib / Blender 2D

游戏机制数据



- 道路资产需要4种模式
  - **地面 (ground)、高架 (elevated)、隧道 (tunnel)、隧道洞口 (slope)**
- 道路资产的属性
  - 总宽度, 机动车/自行车/人行道位置
  - 道路 prop：路灯、行道树，公交站亭等
  - 游戏中造价和限速
  - 每个 mesh 对应的模式和材质类型
  - 特殊的路口 node mesh（右图）
- **离线生成最终输出：**
  - 道路 Mesh (FBX)，资产属性文件 (XML)

运行时支持



- FBX 和 XML 使用游戏内置的资产编辑器封装
  - 资产编辑器没有脚本接口，需实现**自动控制和监听UI**，处理导入队列
- 材质：
  - 所有资产**共用一套材质集**，材质在运行时用单独的容器资产加载
  - 封装的道路资产只含Mesh，节省大量存储空间和载入时间
- 游戏内补丁
  - CSUR 对车道的处理和原版游戏存在大量区别，需**重写游戏机制的大量方法**
  - Harmony 库： C# 运行时方法装饰和替换
  - 《城市：天际线》mod 开发的基本操作，有兼容性和崩溃风险

目录

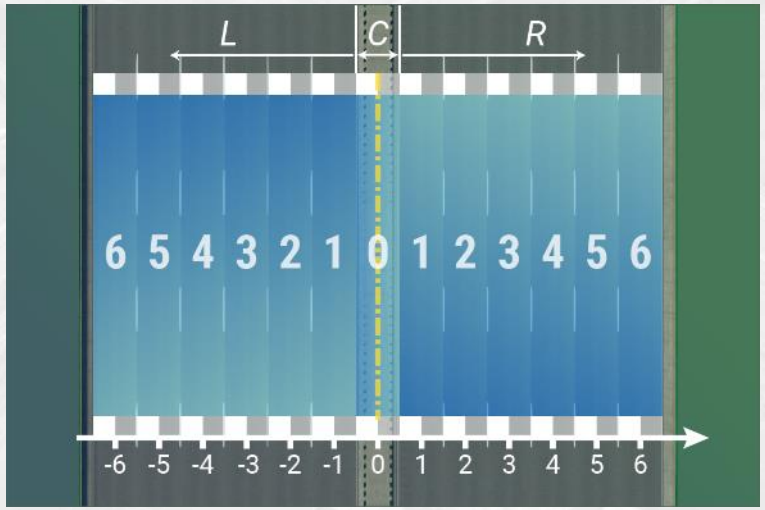
用户评价

- **难学！**修路比原版游戏还慢
  - 内容和操作复杂，学习曲线非常陡，熟练后才能高效使用
- **难用！**到处都是“bug”
  - 覆盖大量原有的游戏机制，同时和部分游戏机制冲突
- **占内存太多了！**电脑根本带不动
  - 离线生成仍然会产生大量的静态资产
  - 枚举计算出的道路模块有 600 多种，一个玩家只会用到一小部分

- 现阶段的《城市：天际线》
  - 对道路模型的精确调整和程序化标线绘制工具趋于成熟，能够满足小规模仿真交通建设的需求
  - **CSUR 逐渐小众化，退出历史舞台（已停止开发）**

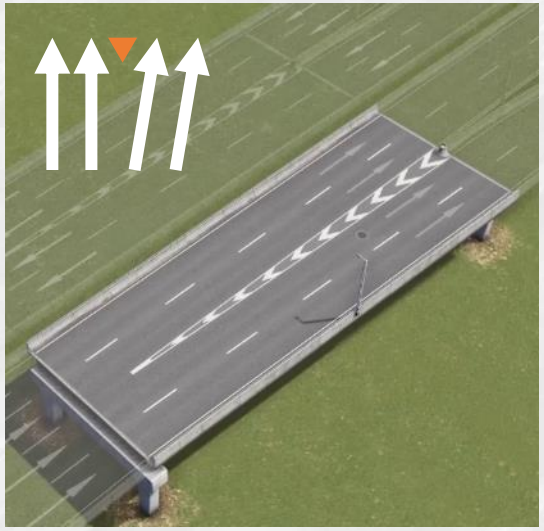
主要设计缺陷

- **绝对车道位置**
  - (相对游戏机制进行的妥协)
  - 1-2位置的二车道和3-4位置的二车道需要制作两个单独的资产
  - 模型之间仅为平移关系，浪费空间和加载时间
  - 同样的一条路有不同的偏移位置，玩家难以理解
  - 现实道路不一定符合车道绝对位置的概念



主要设计缺陷

- **全离线生成**
  - 设计初衷：三维美术/资产制作批处理工具
  - CSUR 在游戏运行时不进行任何几何和材质处理
  - 枚举的资产范围有限 (~600)，理论存在的道路组合有上千种，玩家经常抱怨在 UI 中找不到资产
  - 无法动态实现更丰富的道路变化，如匝道分岔角度的改变



600多种离线资产中没有包括从5号车道移动到7号车道的组合

离线生成 vs 在线生成

本质：时间复杂度和空间复杂度的权衡

- [✔] GPU 瓶颈时对性能影响小 (例外：geometry shader)
- [✔] 便于开发和调试，游戏内调试 (尤其二次开发) 复杂

- [✘] 消耗大量 RAM 和存储空间，拖慢载入时间，RAM 瓶颈时影响性能
- [✘] 无法生成随机性 (stochastic) 和无限大的场景，程序化生成的结果是静态的

《城市：天际线》仿真建造的“阿喀琉斯之踵”：玩家订阅大量(可达1万)资产，游戏一次性载入全部，导致内存和显存成为最主要的性能瓶颈。

CSUR的大量内容进一步增加消耗导致内存溢出，上小时的等待游戏载入功亏一篑！

目录

城市模拟游戏的未来

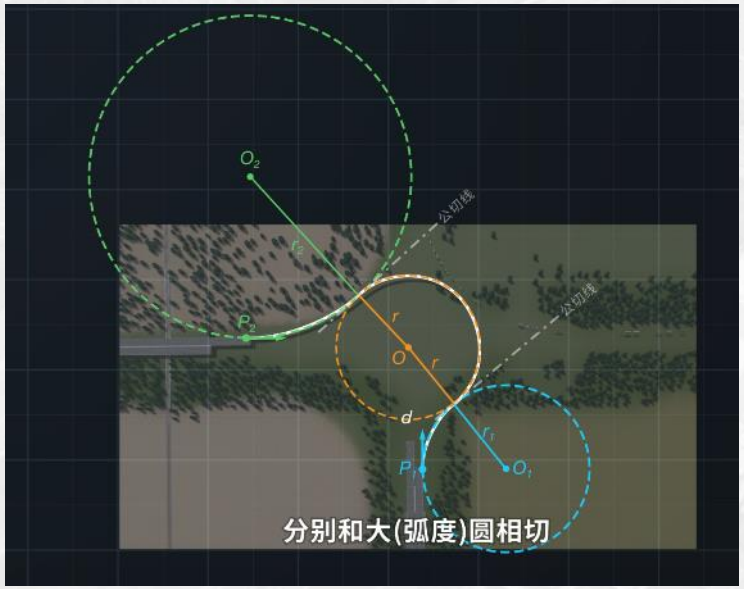
- **大规模动态场景的实时渲染**
  - 城市模拟游戏的渲染质量落后静态场景游戏 5 年或以上
  - 难以根据特定场景优化（如静态LOD），无法烘焙光照
  - 资产即时加载 + 大量 Mesh 渲染 + 实时全局光/光线追踪 (Unreal Engine 5)
- **城市交通、人口、产业的实时模拟**
  - 无外部资产和 mod 的《城市：天际线》主要为 CPU 瓶颈 (30 FPS @ Ryzen 9 5950X)
  - 高性能 Agent-based 模拟算法
- **城市模拟的社会意义**
  - 游戏的体验和参与度：众筹虚拟城市场景
  - 数字孪生云平台

道路场景的“三维”

- **易用 User-friendliness**
  - 易用性存在不同需求：场景编辑工具—美术；游戏道路工具—玩家
  - 道路之间和道路与城市其他元素的交互：道路过渡/交叉口，道路对地形的影响，道路和建筑物之间的连接
- **丰富 Diversity**
  - 道路组成的丰富性：车道数量，人行道，快速公交(BRT)，有轨电车，无轨电车，…
  - 道路过渡：车道加减、不同角度的分流、合流
- **正确 Correctness**
  - 虚拟道路近似或完全符合工程技术标准
  - 手工制作越多，正确性越难以保证（玩家或游戏美术不一定熟悉道路设计）
  - 正确性能增加虚拟世界的沉浸度，但不一定适用于所有场景（如赛车游戏车道比较宽）

道路场景的“三维”

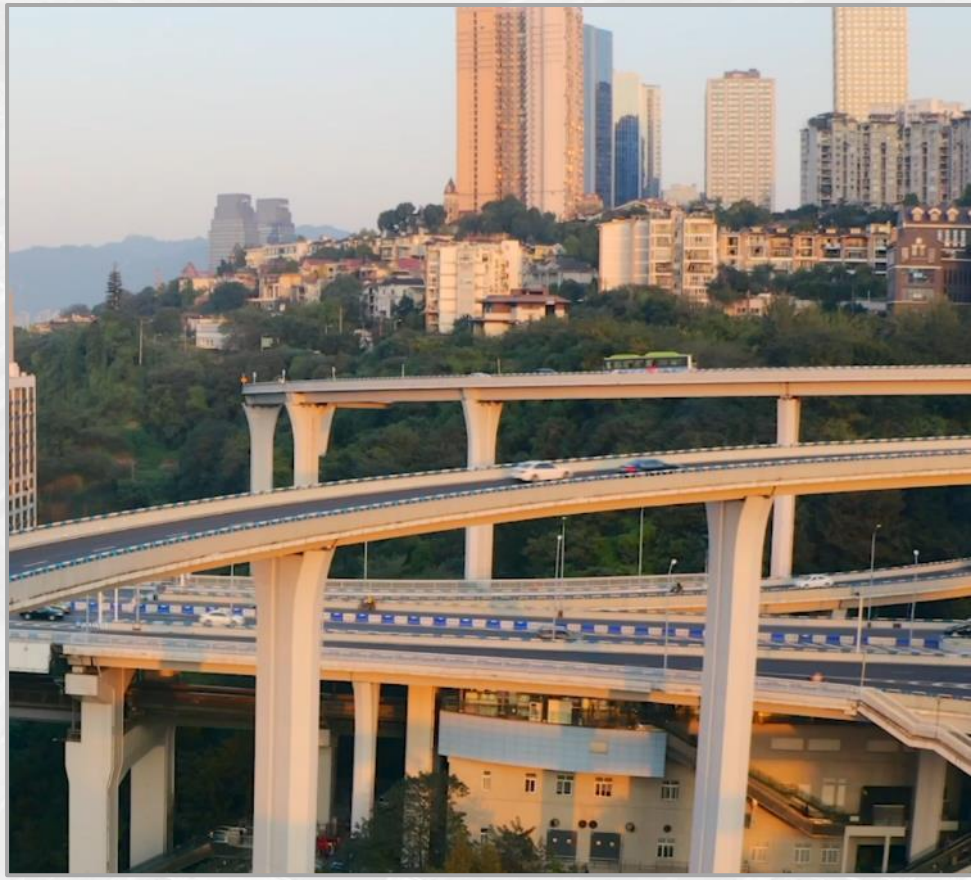
- 实例：**高级道路工具**（CSUR衍生项目）



- 易用：拖动圆心可以动态修建整条立交桥匝道，**比较易用**
- 丰富：CSUR 提供**相对充足**道路的丰富性
- 正确：通过圆弧生成贝塞尔曲线的顶点位置，比**玩家手工修路准确，但仍不符合现实道路**（现实世界转弯半径不会突变）

道路线形生成算法

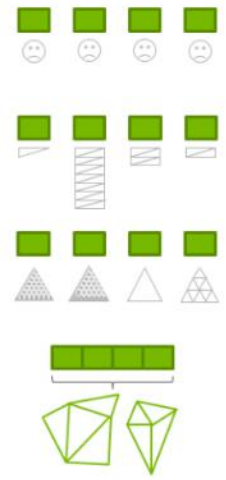
- **易用和正确的兼得同样需要靠程序化生成**
- **输入：**
  - 道路路段端点位置和水平方向，道路曲线所需圆的位置和半径，道路之间的连接
- **输出：**
  - 道路的精确水平曲线，圆和圆/直道路段之间**的曲率半径连续**
  - 道路的斜坡坡度变化 (纵曲线)，**上/下坡需平滑过渡**
  - 道路的倾斜程度 (超高)，**车速越快转弯半径越小，道路倾斜越多**
  - 动态的车道宽度，**转弯处车道需加宽**保证安全性

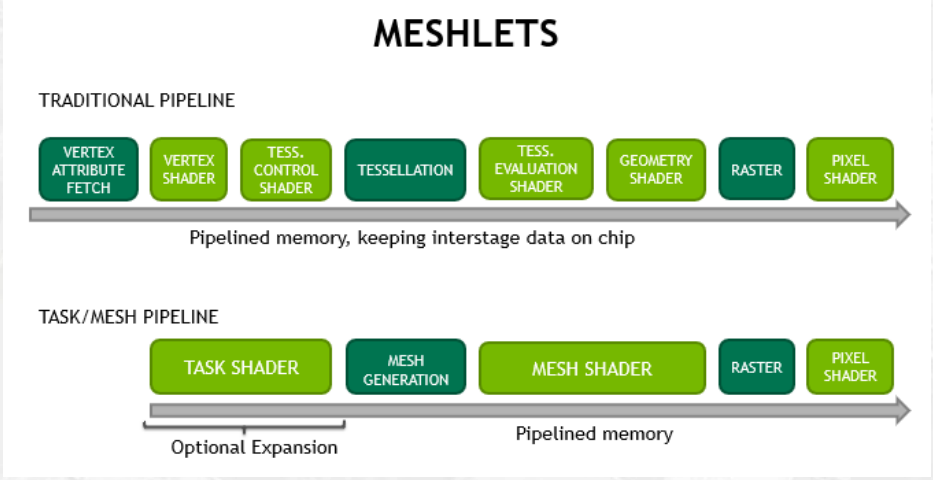


程序化渲染

- Procedural **rendering**
  - 用**渲染管线**进行程序化生成
- Mesh shader (NVIDIA Turing / AMD RDNA2)
- CPU 不处理 Mesh 的全部几何信息，只提供 Mesh 的生成参数

Shader		Thread Mapping	Topology
Vertex Shader	No access to connectivity	1 Vertex	No influence
Geometry Shader	Variable output doesn't fit HW well	1 Primitive / 1 Output Strip	Triangle Strips
Tessellation Shader	Fixed-function topology	1 Patch / 1 Evaluated Vertex	Fast Patterns
Mesh Shader	Compute shader features	Flexible	Flexible within work group allocation

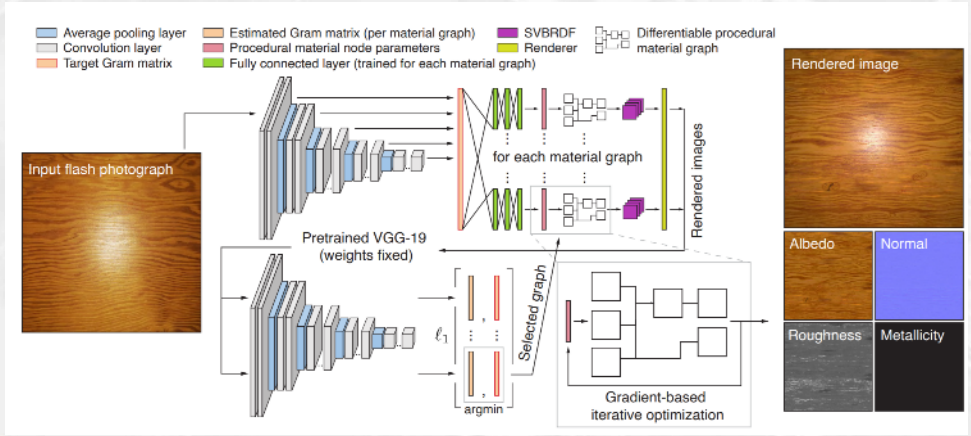




可微分程序化生成

- 可微分生成 (differentiable generation)
- 从最终实现效果出发，端到端构建生成工作流 (如材质的 node graph)
- 相比黑盒式的机器学习算法更易于控制和解释

- **场景生成能否用同样的方法?**



L. Shi *et al.*, MATch: Differentiable Material Graphs for Procedural Material Capture, *ACM Trans. Graph.* **2020**, 39(6) (SIGGRAPH Asia 2020), 196.

致谢

CSUR 开发团队

Amamlya 策划/美术
steamcommunity.com/id/43187386

PCFantasy 程序 (运行时)
github.com/pcfantasy

CSUR 支持团队

我家是个大鱼缸
slicky.fun

西伯利亚挖出一只猫
space.bilibili.com/34956752

LittleGolden
github.com/Littlegolden

paradox INTERACTIVE

COLOSSAL ORDER