InfermonSAGA Dev Report

Zhang Jinrui*

jerryzhang40@gmail.com

20250124

摘要

In this article, the development processes and plans are logged.

1 Main Goal

The inspiration is from [4, DragonDungeonRun], which is a very simple to implement grid base game.

But I don't want it just to be a vertical level only game. But more rather like a open world game, with systems like "farming", "fighting", "adventrue" and even some kind of character "RPG" and "progression".

For the "adventure" part, the world is just a lot of square grid, which the player is always takeup one grid. The grid can be viewed as little piece of a manifold. The map of the world can be a lot of 2D-manifold with different inherent (nP^2 or nT^2, does't mater afterall we will have a more realistic and non-theoretical method to realise this in a game), connected by "portal". This forms the core "adventrue" part of the game.

For the "fighting" part is just like usual RPG game, the monster or enemies also takeup one square grid, and less dynamic are needed for the moster AI.

For the "farming" part we can use the tools to make a square grid in to a farmland,which we can grow agricultures on this square grid. The productions by farming can help "exploration" and "fighting"

*alternative email:zhangjr1022@mails.jlu.edu.cn

The final part is mainly need high quality character design. This need a lot of art and game script effort, which may not be affordable for myself at present.

2 Roadmap

2.1 Topology structure for adventrue

For this we need a mathematic based data structure to save our map and portals.

Take the one eighth of a sphere as an example. Figure 1 shows how this expansion will look in general.

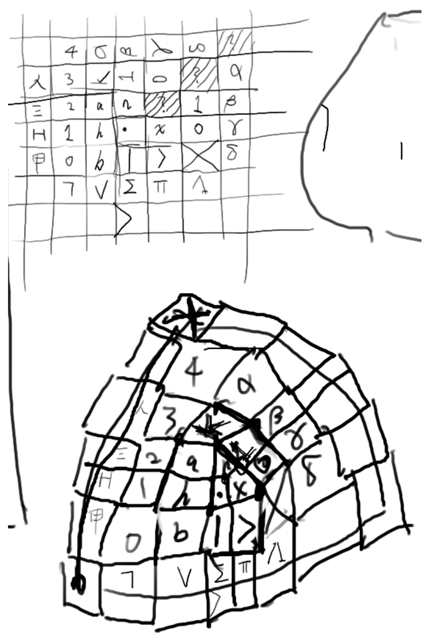

图 1: Suqare tile expansion of $\frac{1}{8}$ sphere

The shadow area with a question mark, means this place have a conflict and will be view as a "Portal".

Definition 2.1 (Portal)**.** The shadowed question marked tile is a portal.

This is inevitable because the curvature of the sphere is not equal to the plane. But this is a flavor of the Infermon view the world he lived in. When you try to enter the "portal" you will end in an another square the current square you leave connected to.

The second example is about a important operation to manipulate of the space Topology. Which is Tunneling. This process is showed in Figure 2. In this process, tile "A" was turned into a portal tile.

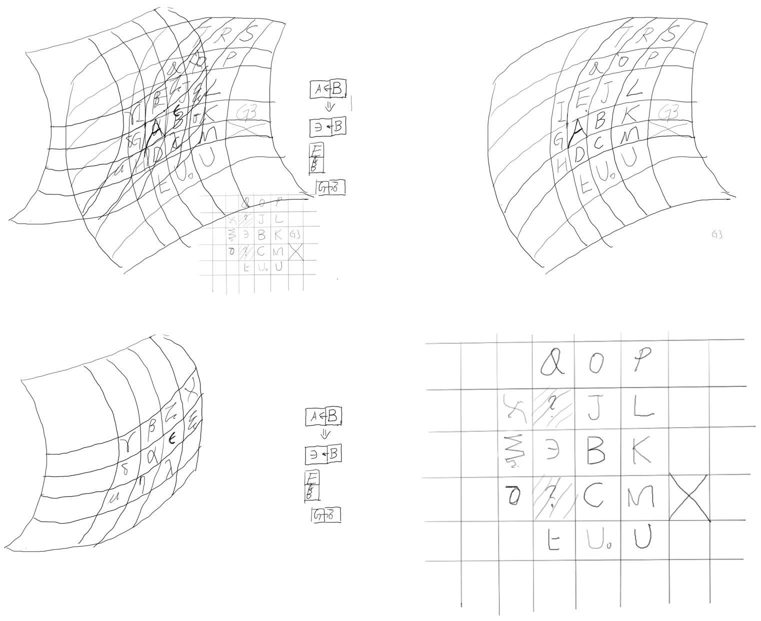

图 2: Tunneling process

For there is some special case for the portal, which is the Tunneling portal, we need to make some more definition. The Tunneling portal has a different visualizaiton requirement.

Definition 2.2 (Intrinsic Portal)**.** Those portal that inevitable to create, when using the ordinary way to expand the Suqare tilling.

Definition 2.3 (Artificial Portal)**.** Those portal that made by the Tunneling process, surround by a 8-tile valid loop.

The whole world is just a single map, the map is maintained in a single list. But the world can be a several disconnected, space bubble.

A block should save four adjacent block. When render it to a Square tile expansion to a 2d-gird. inevitablly some Intrinsic Portal will be created.

Another Portal type is ArtificialPortal which treat as a object on a space block. The ArtificialPortal have linked two space block together, and if one goes into one from one side, he will come out in the conjugate space block from the other side.

Tunneling process is a advanced technique, for the player have some extra level in the game, he would be modify the link of the adjacent relationship of a single block.

Now we will be focous to realise this data structure

2.1.1 list

The following is a implementation GPT give. This list is for save the map data in a linked list. The Suqare tile block is save in the list.

```cpp
#include <iostream>
#include <list>
#include <memory>
#include <fstream>
#include <string>

template <typename T>
class MyList {
private:
    std::list<std::unique_ptr<T>> elements;

public:
    // 添加元素
    void push_back(std::unique_ptr<T> element) {
        elements.push_back(std::move(element));
    }

    // 删除第一个元素
    void pop_front() {
        if (!elements.empty()) {
            elements.pop_front();
        }
    }

    // 遍历并打印元素
    void print_all() const {
        for (const auto& element : elements) {
            element->print();
        }
    }

    // 保存到文件
    void save_to_file(const std::string& filename) const {
        std::ofstream outFile(filename);
        if (!outFile) {
            std::cerr << "Error opening file for writing." << std::endl;
            return;
        }

        for (const auto& element : elements) {
            outFile << *element << std::endl;
        }

        std::cout << "List saved to file: " << filename << std::endl;
    }

    // 从文件加载
    void load_from_file(const std::string& filename) {
        std::ifstream inFile(filename);
        if (!inFile) {
            std::cerr << "Error opening file for reading." << std::endl;
```

```cpp
            return;
        }

        elements.clear(); // 清空当前链表
        while (!inFile.eof()) {
            auto element = std::make_unique<T>();
            if (inFile >> *element) {
                elements.push_back(std::move(element));
            }
        }

        std::cout << "List loaded from file: " << filename << std::endl;
    }
};

// 示例数据结构
struct DataBlock {
    int id;
    std::string name;

    DataBlock() = default;
    DataBlock(int id, const std::string& name) : id(id), name(name) {}

    void print() const {
        std::cout << "ID: " << id << ", Name: " << name << std::endl;
    }

    // 序列化到文件
    friend std::ostream& operator<<(std::ostream& os, const DataBlock& block) {
        os << block.id << " " << block.name;
        return os;
    }

    // 从文件反序列化
    friend std::istream& operator>>(std::istream& is, DataBlock& block) {
        is >> block.id >> block.name;
        return is;
    }
};

int main() {
    MyList<DataBlock> myList;

    // 添加元素
    myList.push_back(std::make_unique<DataBlock>(1, "Block1"));
    myList.push_back(std::make_unique<DataBlock>(2, "Block2"));

    // 打印链表
    std::cout << "Initial list:" << std::endl;
    myList.print_all();

    // 保存到文件
    myList.save_to_file("data.txt");

    // 从文件加载
    myList.load_from_file("data.txt");

    // 打印加载后的链表
    std::cout << "After loading from file:" << std::endl;
    myList.print_all();

    return 0;
}
```

2.1.2 Square tile

2.2 fighting system

2.3 farming system

3 Dependencies

This section, the third-party dependencies will be logged and explained about how they are used in this specific project.

3.1 GLAD

The glad is used for the version that support the multiple-window graphics. The configuration for this dependence is according to [1, glad-MultiwinMx]. We need to do this once in the top CMakeLists.txt.

```
set(GLAD_SOURCES_DIR "${PROJECT_SOURCE_DIR}/glad")
add_subdirectory("${GLAD_SOURCES_DIR}/cmake" glad_cmake)
glad_add_library(glad REPRODUCIBLE MX API gl:core=4.3)
```

Remember to include glad right before glfw as follow.

```
#include <glad/gl.h>
#include <GLFW/glfw3.h>
```

3.2 GLFW

The main graphics dependencies is glfw3. I used its sources code form its Github repository.

And Simply use this two line in CMakeLists.txt is okay.

```
add_subdirectory(glfw)
include_directories(glfw/include)
```

Then include GLFW in the file use it.

```
#include <GLFW/glfw3.h>
```

3.3 IMGUI

I choose to use the maximum features of the imgui, which contains ”docking” and ”Multi Viewports”

To set up imgui with the glad and glfw is according to [2, setupImgui].

To set up imgui with ”Multi Viewports” feature is according to [3, text].

参考文献

[1] https://github.com/Dav1dde. glad multiwin mx. `https://github.com/Dav1dde/glad/tree/glad2/example/c++/multiwin_mx`. Accessed: 2025-01-24.

[2] https://github.com/Dav1dde. imgui for glfw opengl. `https://github.com/ocornut/imgui/wiki/Getting-Started#example-if-you-are-using-glfw--openglwebgl`. Accessed: 2025-01-24.

[3] https://github.com/ocornut. imgui multi viewports. `https://github.com/ocornut/imgui/wiki/Multi-Viewports`. Accessed: 2025-01-24.

[4] Nikke. Dragon dungeon run. `https://nikke-goddess-of-victory-international.fandom.com/wiki/Dragon_Dungeon_Run`. Accessed: 2025-01-24.