Unity3d + Flyweight

ОПТИМИЗАЦИЯ 2D КАРТЫ

Привет меня зовут Андрей

План

- Проблемы и возможности Unity3d
- Построение и проблема оптимизации 2d карты
- Оптимизация с использованием шаблона Flyweight
- Тесты
- Демо
- Викторина
- Итоги

Unity3D

Страшилки Unity3d - разрушение мифов и легенд

Страх	Действительность
Игры на Unity тормозят	X
Mono - со всеми вытекающими	X
Нет «game loop»'а – точки входа	X
Местный GUI не подходит для построения сложных интерфейсов	V
Невозможно сравнивать файлы проекта через diff	V
На Unity не делают большие и качественные проекты	?

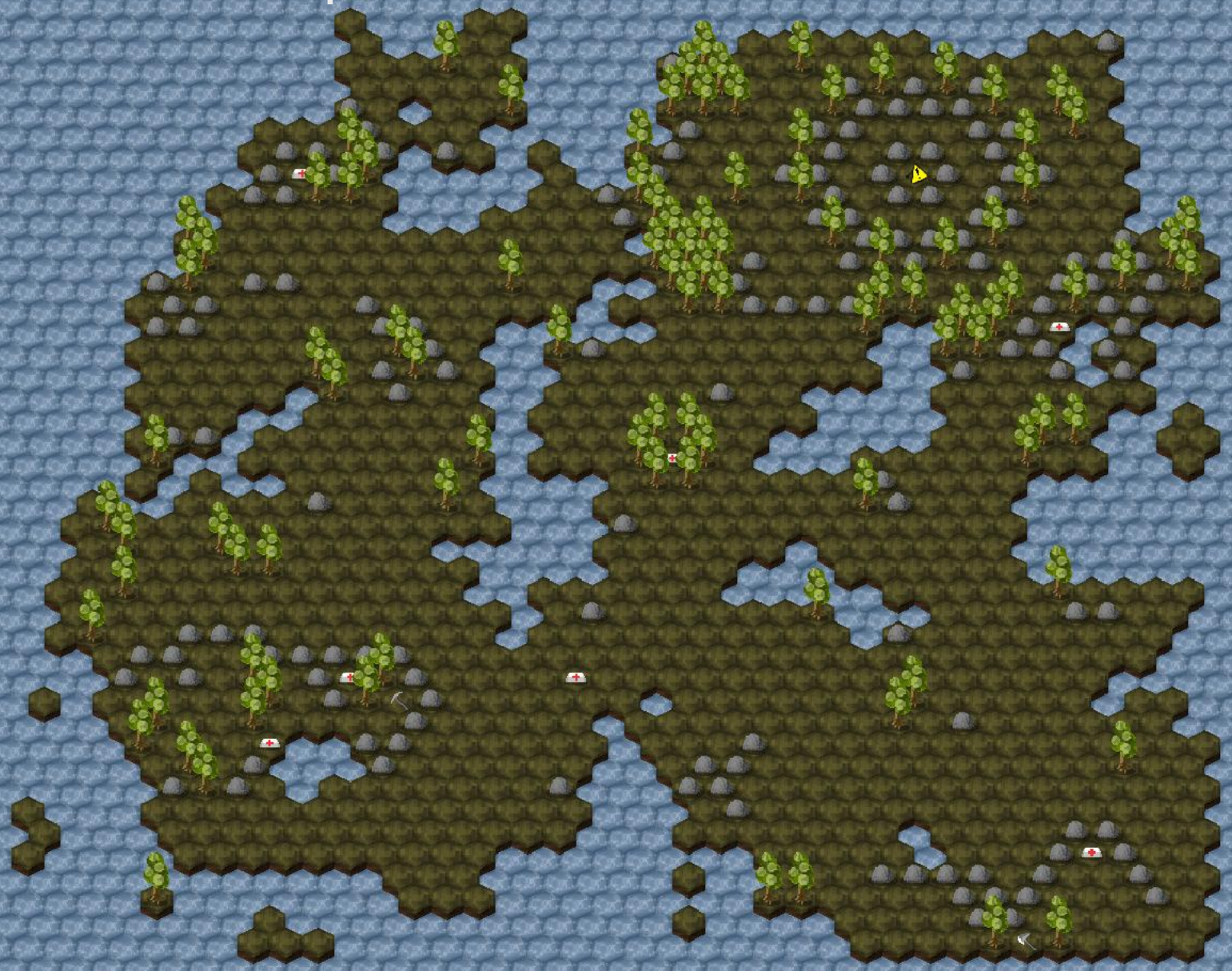
Проекты на Unity за 2018

- ***Aegis Defenders***
- ***Beat Saber***
- ***Dusk***
- ***Forgotton Anne***
- ***Ghost of a Tale***
- *Battletech*
- *Bloons TD 6*
- *Chessaria: The Tactical Adventure*
- *Crowfall*
- *Fe*
- *Florence*
- ***Hollow Knight***
- ***Just Shapes & Beats***
- ***OK K.O.! Let's Play Heroes***
- ***Overcooked 2***
- ***PC Building Simulator***
- *GTFO*
- *House Flipper*
- *Ingress Prime*
- *The Last Night*
- *The Lost Legends of Redwall*
- *Ooblets*
- ***Pillars of Eternity II: Deadfire***
- ***Subnautica***
- ***Two Point Hospital***
- ***Overload***
- ***Return of the Obra Dinn***
- *Pathfinder: Kingmaker*
- *Runner3*
- *Shadowgun Legends*

Возможности Ori - 2014

Трейлер **https://www.youtube.com/watch?v=cklw-Yu3moE**

Построение 2d карты



Результаты запуска

Параметр	Показатель
Карта	100 x 100 – все ячейки анимированные
Объектов на сцене	20 018
Загрузка карты	28.5 с
FPS	11
Расход оперативной памяти	2.85 гб

Анализ проблемы

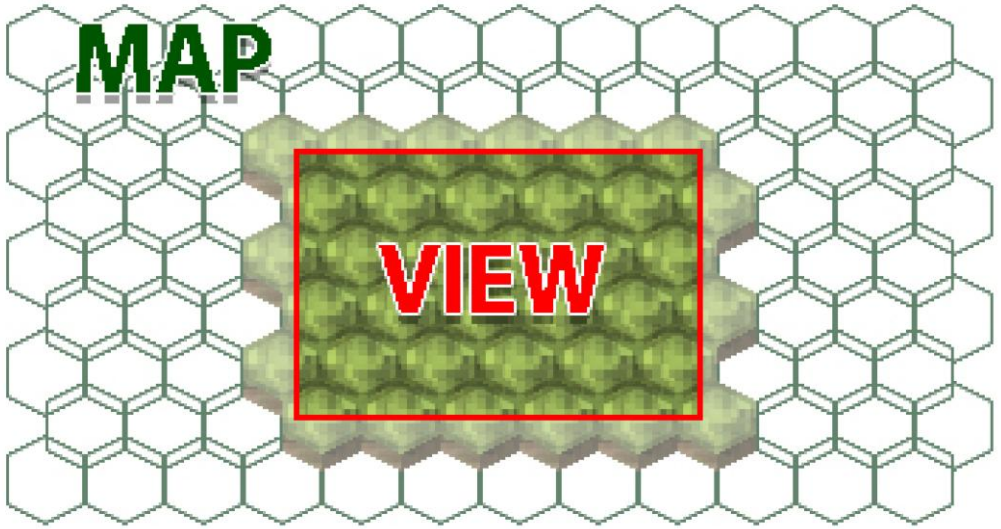
- На сцене присутствует ОЧЕНЬ много ячеек карты, часть ячеек с анимацией.
- Чем больше ячеек на сцене, тем больше расходуется память.
- Ячейки с анимацией нагружают процессор.

Викторина

- **Как можно оптимизировать?**

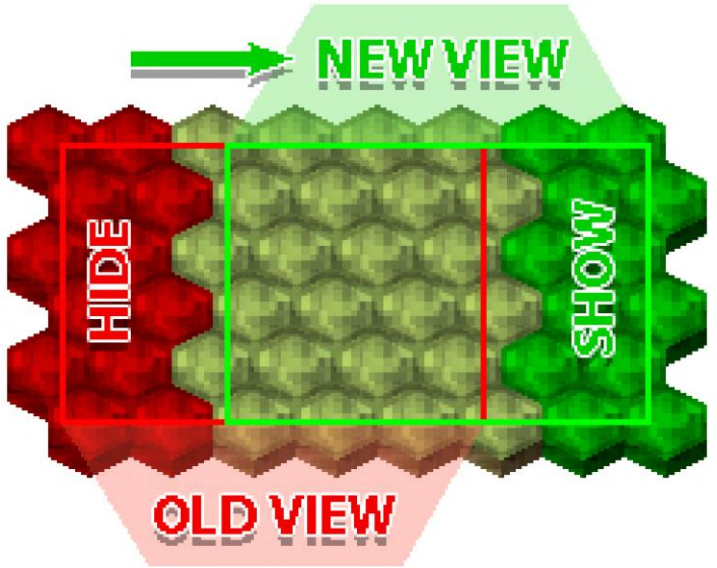
Архитектура решения

- **Решение** - ограничить количество одновременно отображаемых ячеек карты.
- Какую часть карты нужно отображать?
- Отображать только ту часть карты, которую видит игрок.



Архитектура решения

- Отображать только те ячейки карты, которые попадают в прямоугольник камеры.
- Перерисовывать видимую область нужно при перемещении камеры.



Структура оптимизации

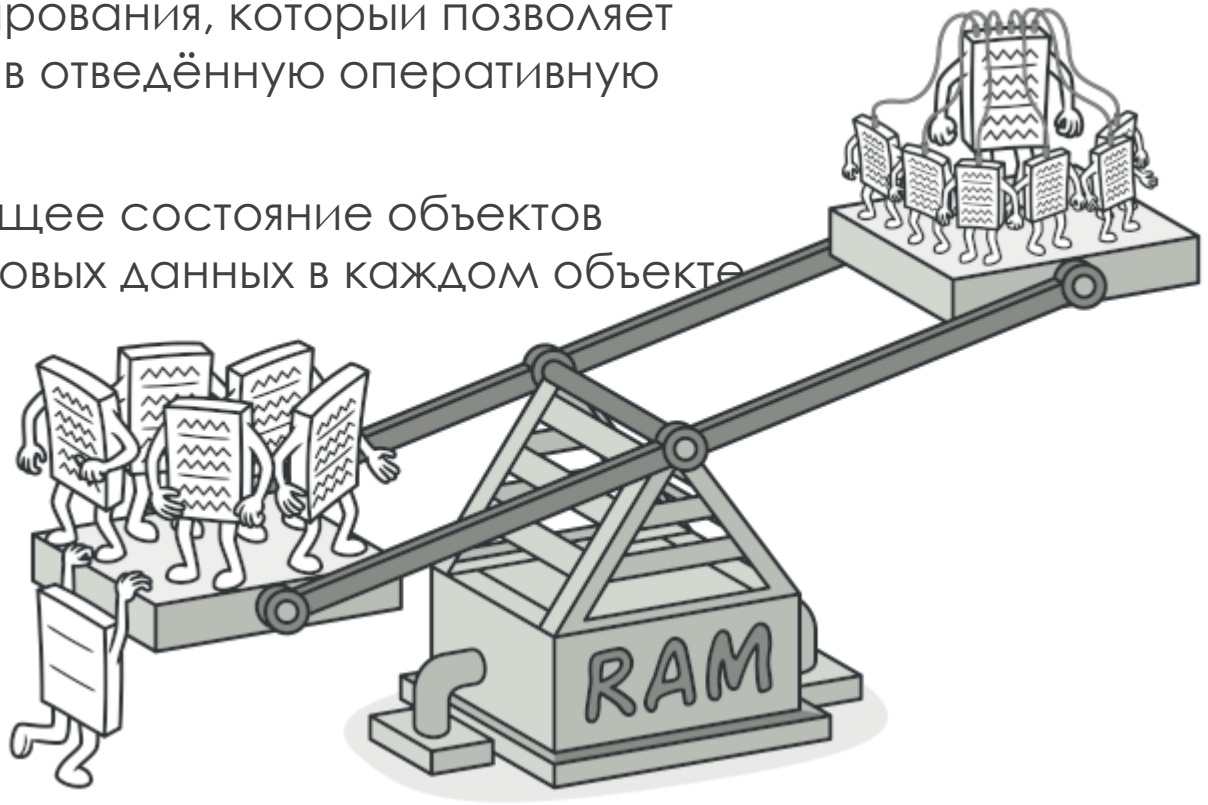
- Назовем объект отвечающий за показ видимой части карты – MapViewer.

Описание работы;

- Отображается некоторая прямоугольная область карты
- Отрисовка области начинается в любой части карты
- При перемещении камеры меняется видимая часть карты

Flyweight

- Flyweight - структурный паттерн проектирования, который позволяет вместить большее количество объектов в отведённую оперативную память.
- Flyweight экономит память, разделяя общее состояние объектов между собой, вместо хранения одинаковых данных в каждом объекте



Flyweight

- Объект реализующий Flyweight будет называться TileFactory – фабрика ячеек сетки.

Описание работы;

- Создает и управляет частями карты
- Предоставляет существующую часть карты или создает новую новую

Викторина

- **Как взаимодействует MapViewer и TileFactory?**

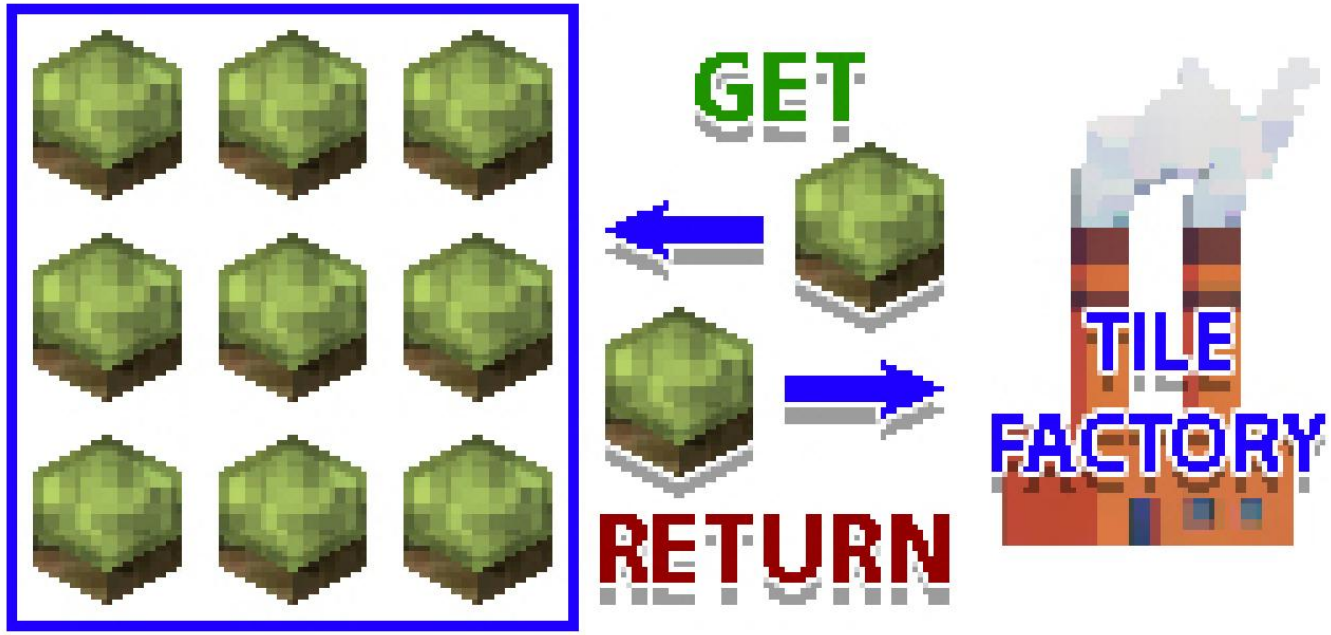
Flyweight

- В системе есть два участника MapViewer и TileFactory

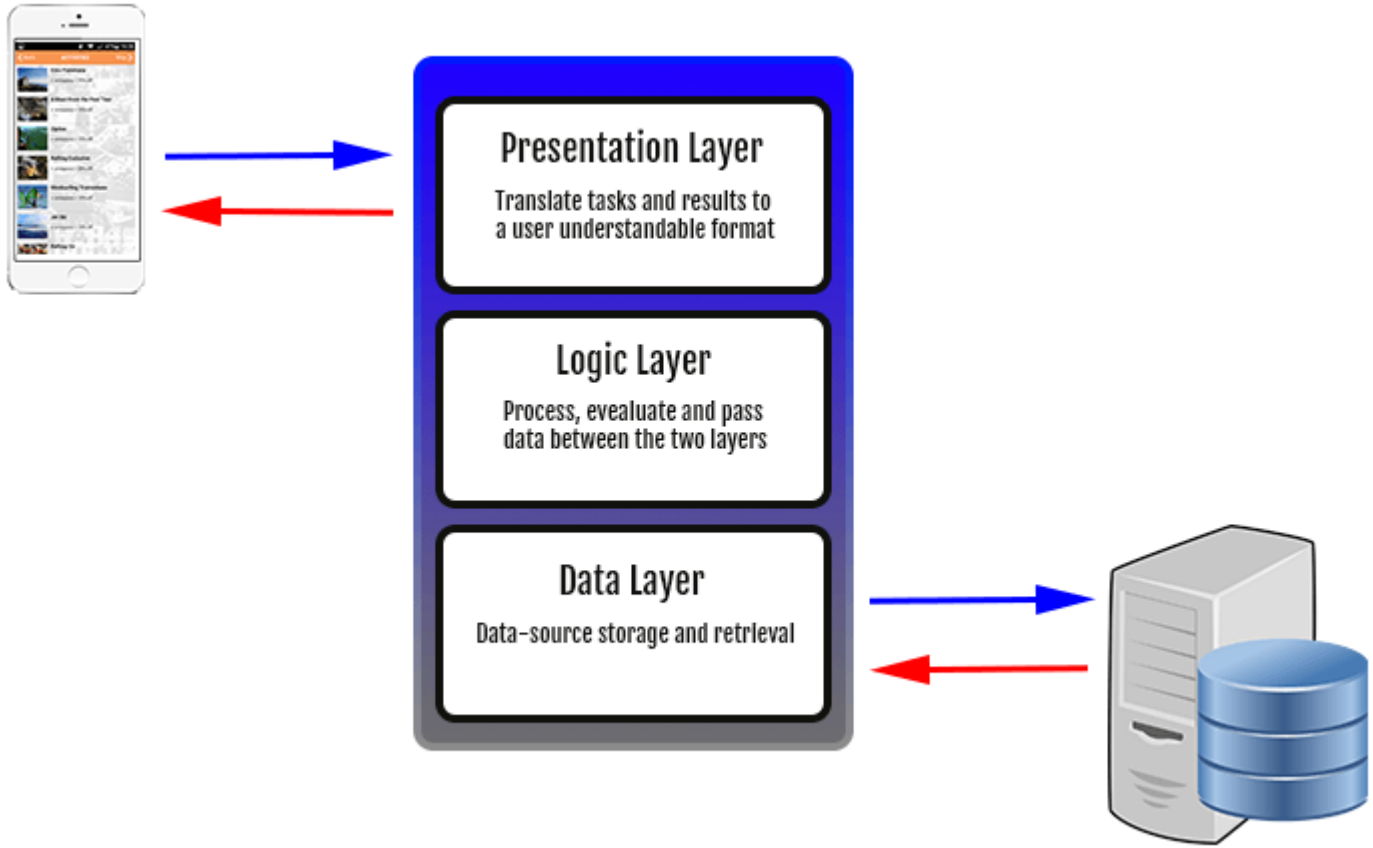
Описание работы:

- MapViewer запрашивает видимые части карты у TileFactory и возвращает скрывшиеся.

Flyweight



Отделение визуальной части карты от логической



Викторина

- **Зачем отделять визуальную часть от логической в рамках задачи оптимизации?**

Отделение визуальной части карты от логической

- Смысл разделения – отделить графику от логики
- Графическую часть можно пере использовать
- Разделение позволит создать минимально необходимую часть графической карты, а все взаимодействия с картой будут проходить через логическую модель
- Что бы не завязывается на конкретные реализации графической и логической частях нужно параметризовать TileFactory и MapViewer.

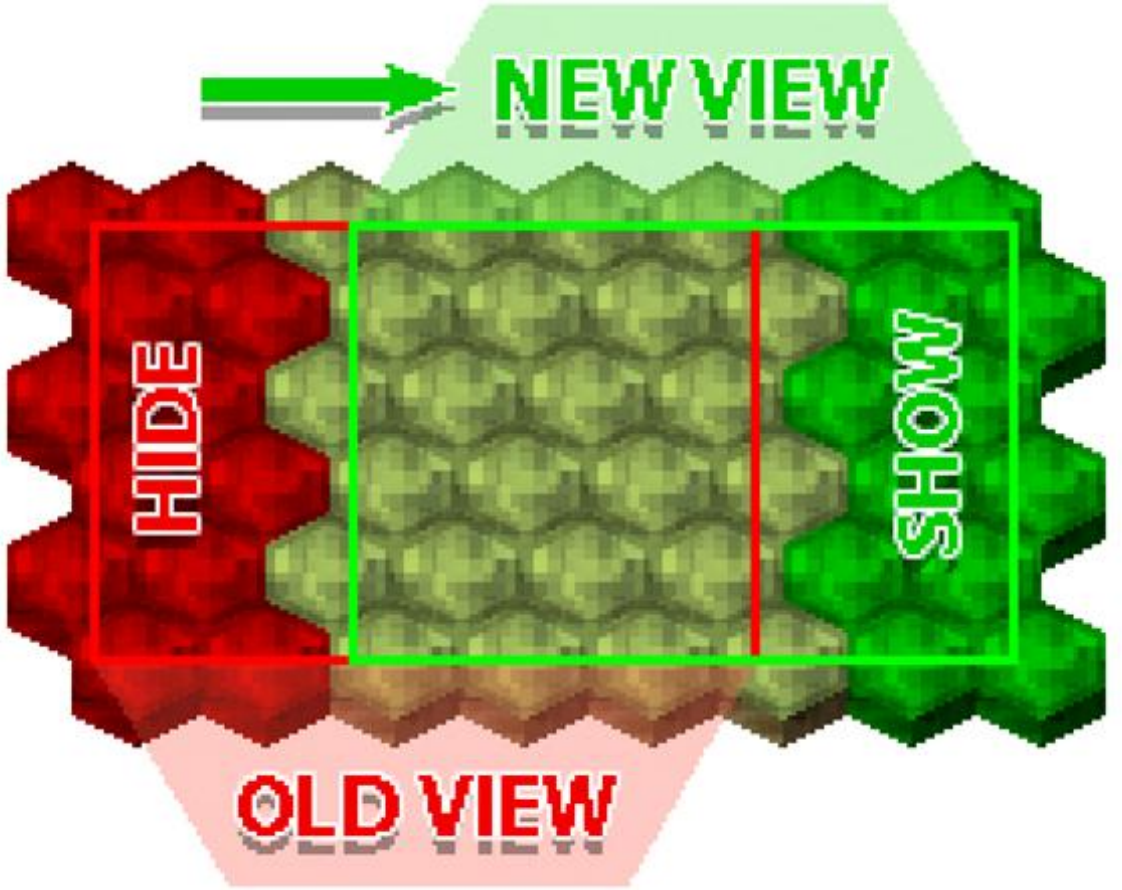
Отделение визуальной части карты от логической

```
//TModel - тип логической ячейки карты
//TTile - тип визуальной ячейки карты
public interface ITileFactory<TModel, TTile>
{
    //Возвращает визуальную ячейку по модели
    TTile GetBy(TModel model);

    //Принимает больше не отображаемую ячейку
    void Return(TTile tile);
}
```

Алгоритм прокрутки

- Проинициализировать видимую часть карты
- Определить смещение
- Прокрутка по горизонтали - скрывшиеся столбцы видимой части карты отправляются в фабрику, а появившиеся запрашиваются у фабрики.
- Прокрутка по вертикали – перемещаются строки.

MapViewer



Подготовка к тестам

- Если с выдумыванием архитектуры, проблем не возникло, то ее реализация была болезненная
- Viwer изначально был написан параметризованным - MapViewer<TModel, TView>. А значит, можно написать тесты!
- Тесты очень сокращают время разработки алгоритма
- В Unity тесты пишутся с помощью NUnit

Тесты инициализации

Данные для тестирования:

- Тестовая карта – двумерный массив
- Точка инициализации MapViewer
- Высота и ширина видимой области
- Текущую видимую область можно запросить у MapViewer

Что можно протестировать:

- По параметрам инициализации можно просчитать ожидаемую область и сравнить ее с областью из MapViewer

Тесты инициализации

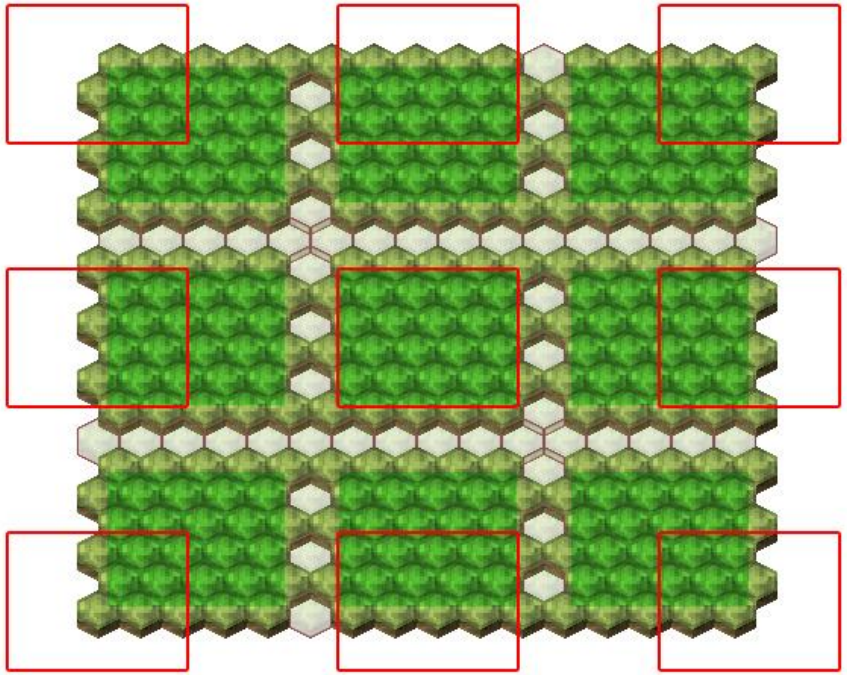
```
/// <summary>
/// Проверка инициализации viwer в нижнем левом углу
/// </summary>
[Test]
public void ViewPortLeftBottomTest()
{
    const int EXPEXTED_MAP_ROW_INDEX = 0,
              EXPEXTED_MAP_COLUMN_INDEX = 0;

    ViewPortInitTest(0, 0,
        EXPEXTED_MAP_ROW_INDEX, EXPEXTED_MAP_COLUMN_INDEX);

    ViewPortInitTest(-1, -1,
        EXPEXTED_MAP_ROW_INDEX, EXPEXTED_MAP_COLUMN_INDEX);
}
```

Викторина

- **Тестов инициализации 9. Почему тестов 9?**

Тесты инициализации

- 8 пограничных состояний и центр



Тесты прокрутки

- Заранее высчитывается количество прокруток при движении из A в B
- Сравниваются обновленные строки и столбцы
- Проверяется перестановка строк и столбцов – пере использование визуальных частей

Оптимизация

- Архитектура позволяет применять разные приемы оптимизации
- Рассмотрим противоположенные приемы оптимизации: по памяти и по процессорному времени

Викторина

- **Часто, оптимизация по памяти обратна оптимизации по процессору. Почему?**

Оптимизация по затратам процессорного времени

VisibilityTilefactory:ITileFactory

- Полная загрузка визуальной части карты
- Управление видимостью частей карты
- Проигрывает по памяти и выигрывает по затратам процессорного времени – мало накладных расходов по построению карты.

Результат оптимизации по процессорному времени

Параметр	Оптимизация
Объектов на сцене	X
Загрузка карты	X
FPS	V
Расход оперативной памяти	X

FlyWeight – оптимизация по памяти

FlyWeightTileFactory:ITileFactory

- Создает ровно столько визуальных объектов, сколько нужно отобразить
- Пере использует созданные объекты
- В памяти хранится столько объектов, сколько нужно отобразить, возрастают дополнительные расходы на восстановление визуальной части по логической, все клетки разные, на клетках могут быть разные объекты: деревья, камни, аптечки и т.д.

Результаты оптимизации

Параметр	Оптимизация	Без оптимизации
Карта	**350 x 350**	100 x 100
Объектов на сцене	**959**	20 018
Загрузка карты	**2 с**	28.5 с
FPS	**60**	11
Расход оперативной памяти	**44 мб**	2898 мб

Демо

Выводы

Unity3d:

- Мощный игровой движок
- Порог вхождения очень низкий
- Разработка хорошо сочетается с классическими шаблонами проектирования – Flyweight
- Поддержка тестов
- Unity – это просто

Выводы

Flyweight:

- Позволяет минимизировать затраты на хранение и обработку множества однотипных объектов – работа с 2d картой.
- Изменения неизбежны!

Ori - 2018

Трейлер https://www.youtube.com/watch?v=kd0zbNw1VOg&t=6s

Контакты

- Хочешь присоединиться к проекту HexCoverry и создавать удивительные миры – пиши мне в телеграмм @InWake