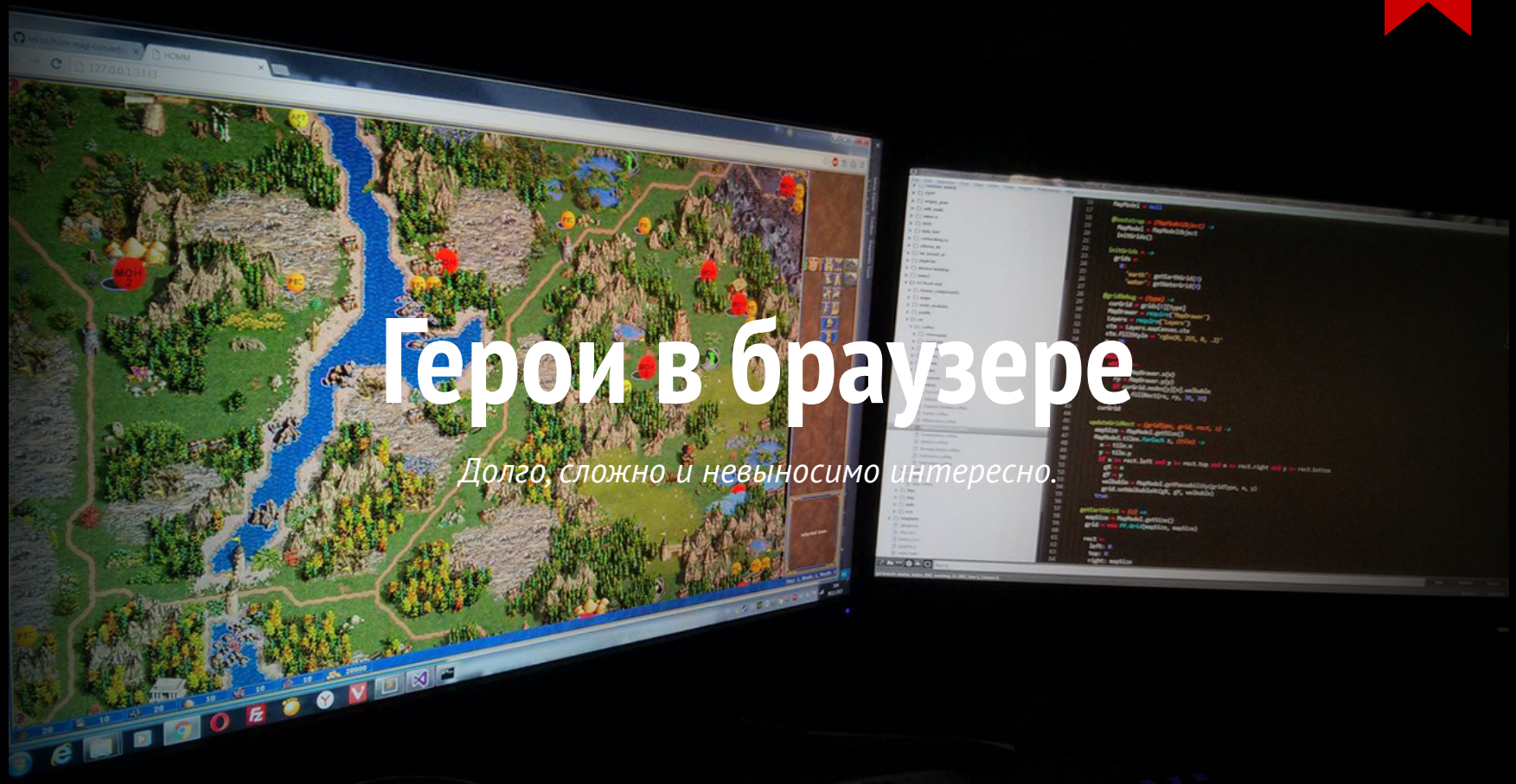


Герои в браузере

Долго, сложно и невыносимо интересно.

Коротаев Александр

1. Работаю в Tinkoff.ru

   Делаю внутренние сервисы и кабинет для бизнеса

2. Хожу на митапы 🤓
3. Делаю SPB-Frontend Drinkcast

   Подкаст о фронтенде за кружкой пива


d
i
n
r
a
c
s
t

Каждые 2 месяца, на гитхабе появляется 1 клон героев

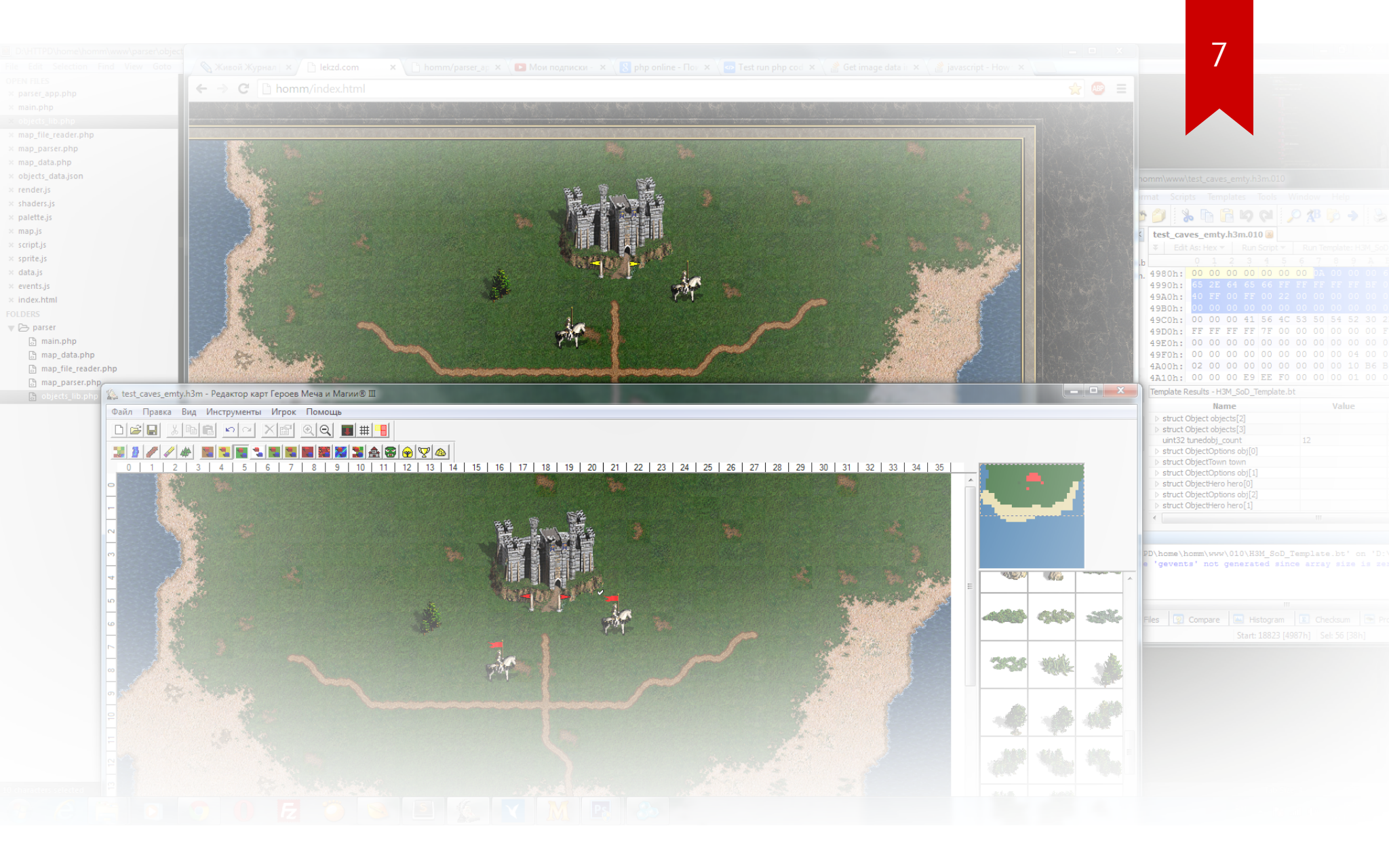
1. sfia-andreidaniel/heroes3
2. mwardrop/HOMM3Clone
3. potmdehex/homm3tools
4. openhomm/openhomm
5. ⚠️vcmi/vcmi

Но никто из них не доделывает до конца и не стремится объединяться

Какие цели ставил

1. Сделать нечто, прыгнуть выше своей головы
2. Заняться чем-то интересным
3. Перестать играть в героев :)

4. сделать красиво


homm/index.html
test_caves_emty.h3m - Редактор карт Героев Меча и Магии® III
Файл Правка Вид Инструменты Игрок Помощь
test_caves_emty.h3m.010
Template Results - H3M_SoD_Template.bt

Из чего состоит игра?

1. Модель данных (персонажи, карта, хранение состояния)
2. Игровой цикл (game loop)
3. Обработка ввода (мышь, клавиатура, джойстик…)
4. Отрисовка (renderer)

А если представить это ввиде кода, то

```
01. const me = {name: 'Alex', left: 0}
02. ...
03. setInterval(() => update(), 1000)
04. ...
05. window.addEventListener('keyup', () => me.left++)
06. ...
07. requestAnimationFrame(() => draw())
```

Модель

```
01. const me = {name: 'Alex', left: 0}
02. ...
03. setInterval(() => update(), 1000)
04. ...
05. window.addEventListener('keyup', () => me.left++)
06. ...
07. requestAnimationFrame(() => draw())
```

Game loop

```
01. const me = {name: 'Alex', left: 0}
02. ...
03. setInterval(() => update(), 1000)
04. ...
05. window.addEventListener('keyup', () => me.left++)
06. ...
07. requestAnimationFrame(() => draw())
```

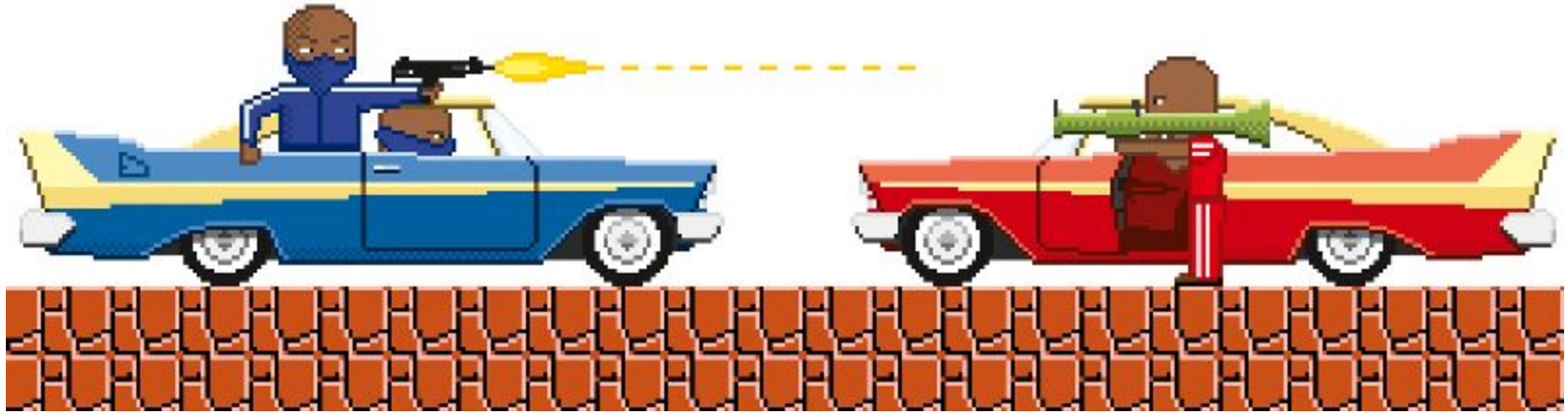
Обработка ввода

```
01. const me = {name: 'Alex', left: 0}
02. ...
03. setInterval(() => update(), 1000)
04. ...
05. window.addEventListener('keyup', () => me.left++)
06. ...
07. requestAnimationFrame(() => draw())
```

Отрисовка

```
01. const me = {name: 'Alex', left: 0}
02. ...
03. setInterval(() => update(), 1000)
04. ...
05. window.addEventListener('keyup', () => me.left++)
06. ...
07. requestAnimationFrame(() => draw())
```

Подробнее про gamedev на JS в книге Сюрреализм на JavaScript

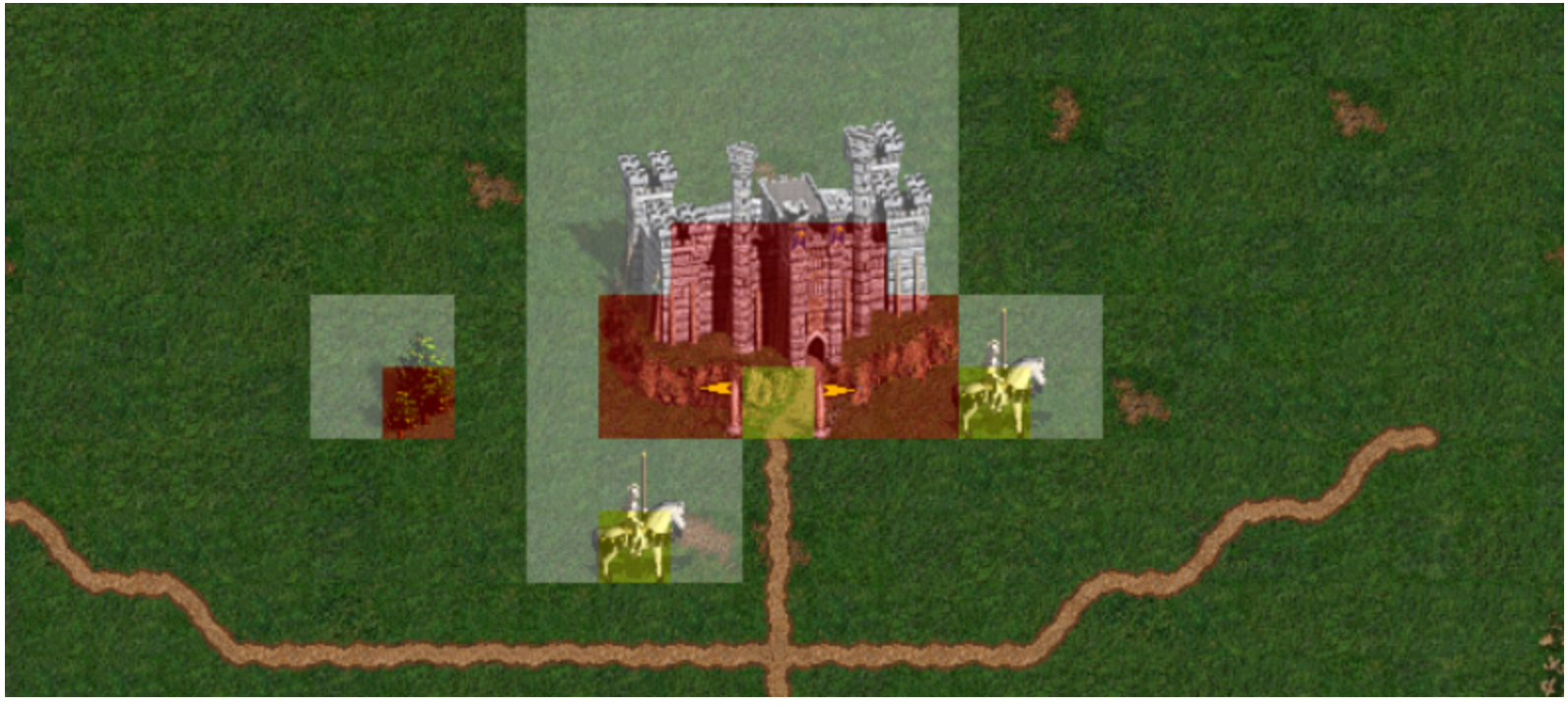


Краткая история разработки

Что мы имеем из входных данных?

1. Оригинальная игра
2. Редактор карт
3. FizMig — большой справочник по всем игровым механикам
4. Много форумов с ребятами, по многу лет копавшимися в "героях"
5. Распаковщик ресурсов 😎

В начале, был рендеринг зеленого поля

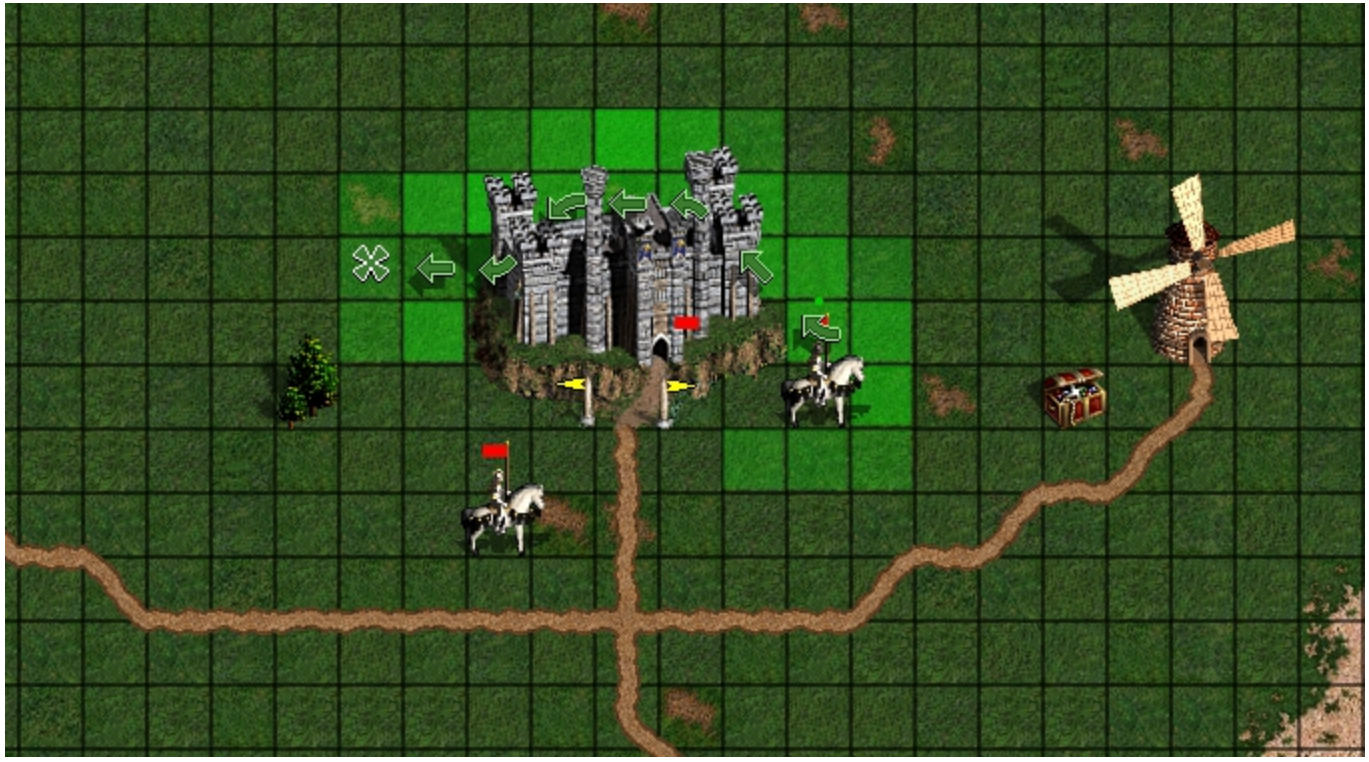


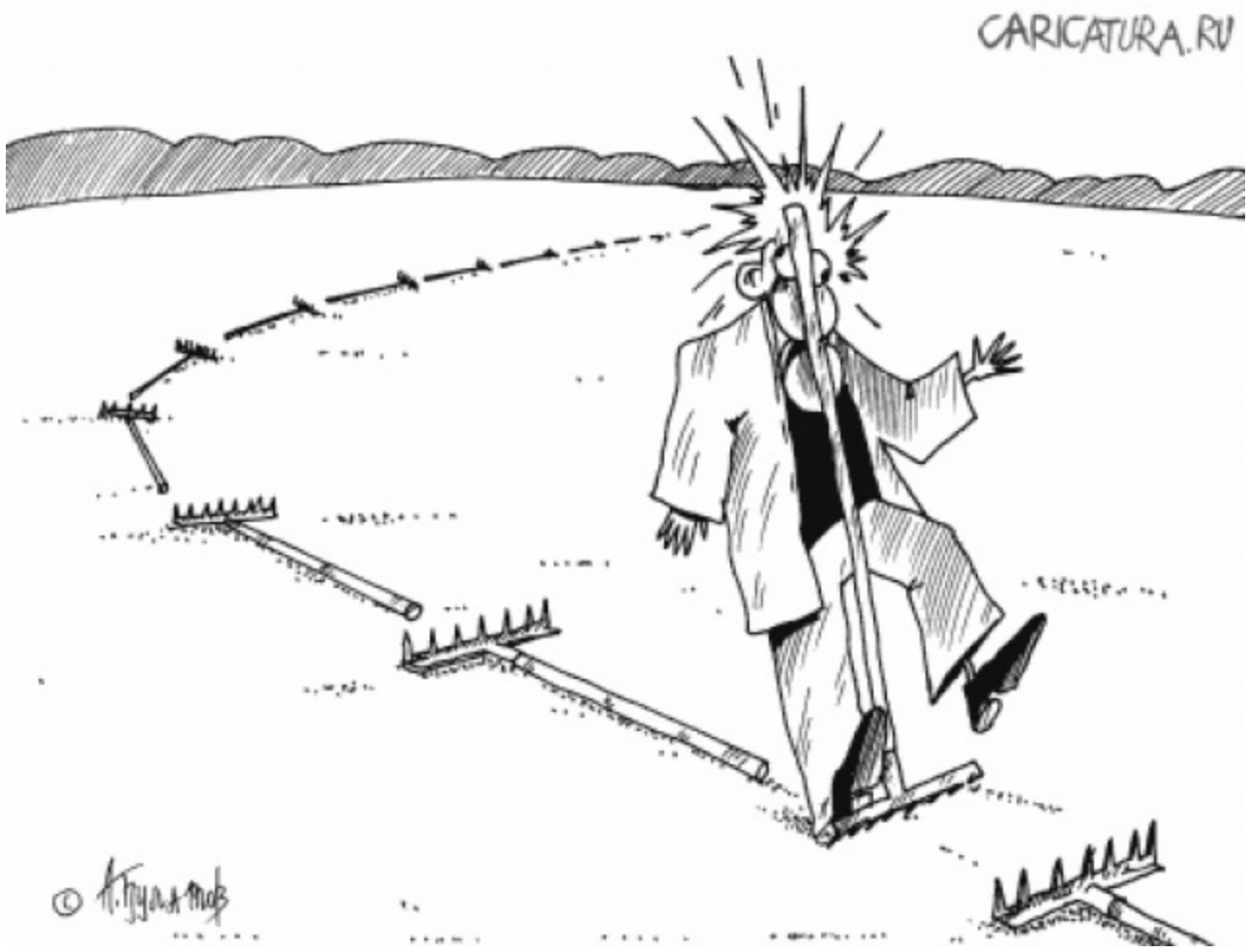
Далее – данные

```
"town": {
    "junk": "int4",
    "owner": "int1",
    "is_name": "int1",
    "name": "str",
    "is_guard": "int1",
    "formation": "int1",
    "is_buldings": "int1",
    "is_fort": "int1",
    "must_spells": "arr",
    "can_spells": "arr",
    "event_quantity": "int1"
},
```

Карты, персонажи, навыки, умения…

Затем — алгоритмы...

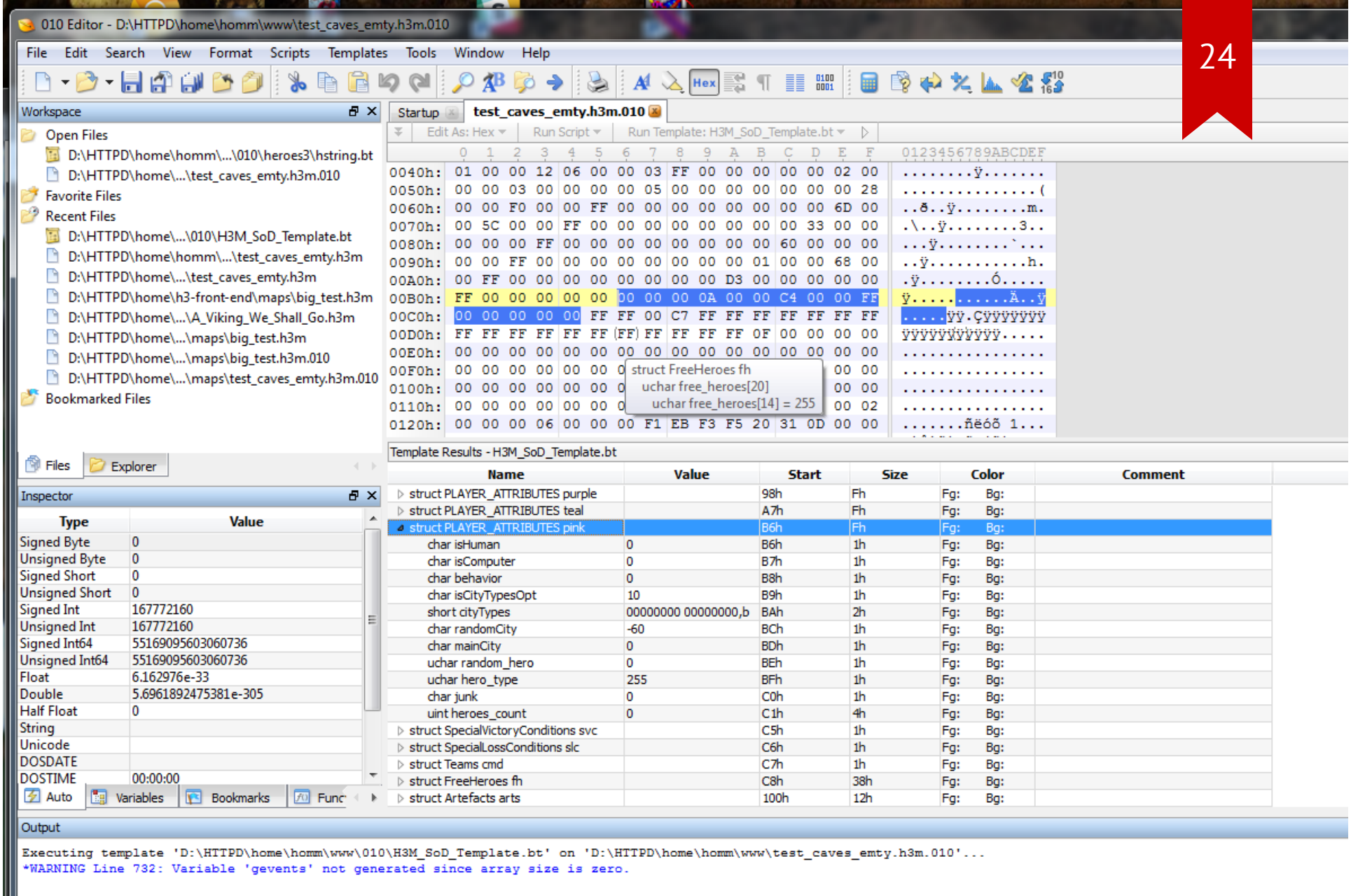



CARICATURA.RU

#1 Парсинг карт


evilfest.h3m - Heroes of Might and Magic® III Map Editor
Файл Редактирование Просмотр Инструменты Игрок: Помощь
ART
RES
ART 4
RES
ART 4
Монстр
Для вызова Помощи - нажмите F1
Нет игрока
108x108x1
Проигрыватель...
65% rogue1.zip
Безымянный - ...
evilfest.h3m - H...
15:15

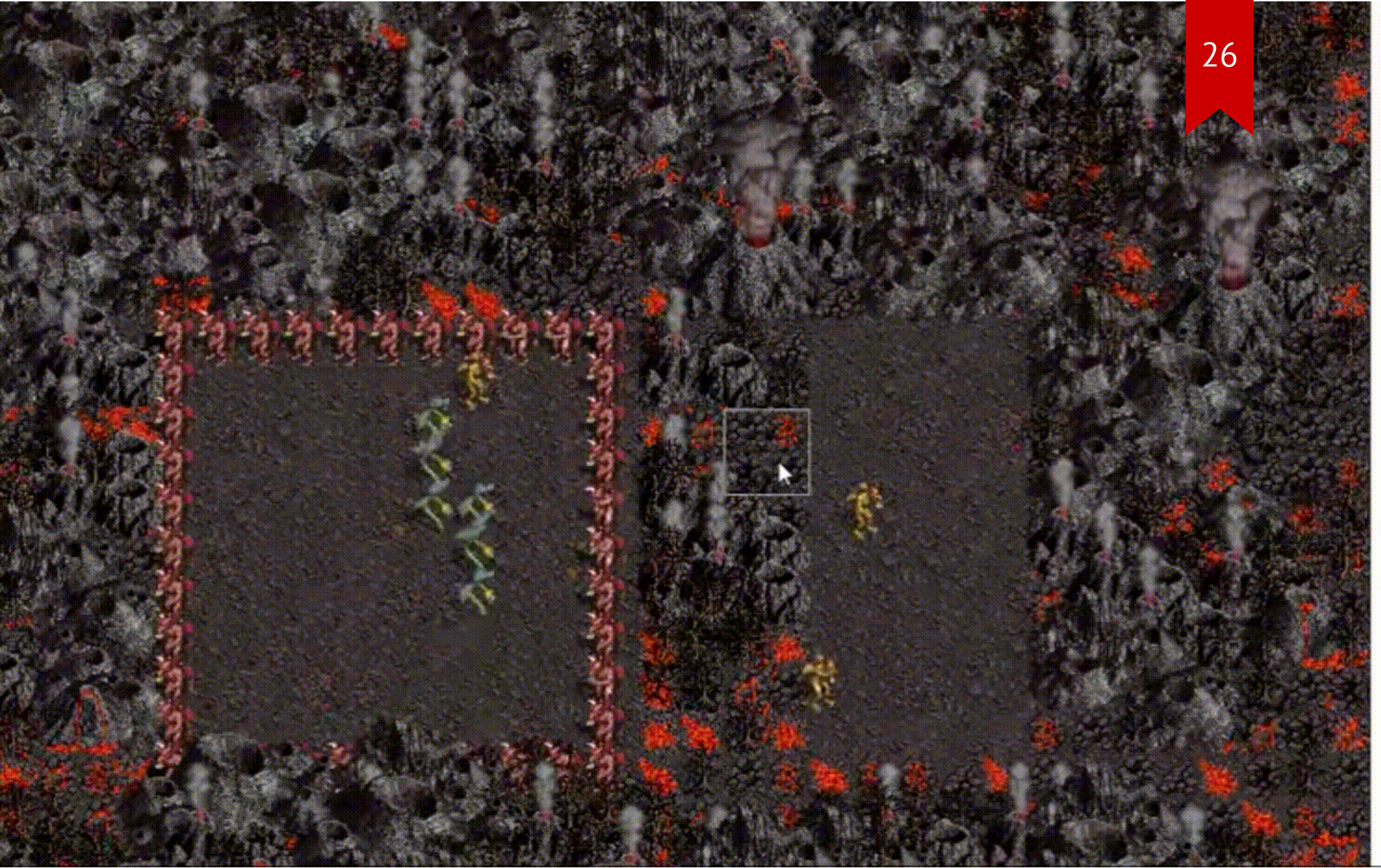
Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	b3	2b	00	3a	35	ba	dd	66	57	7c	24	c1	4f	c9	19	06	³+.:5°ÝfW|$ÁOÉ..
00000010	43	46	d1	31	a7	c5	4b	b8	2f	fe	03	e0	22	61	57	77	CFÑ1§ÅK¸/þ.à"aWw
00000020	24	79	23	dc	21	f6	2c	d4	18	2e	91	c3	b2	67	b5	45	$y#Ü!ö,Ô..'Ã²gµE
00000030	ab	ca	ed	af	02	61	51	0d	4e	ea	1e	87	cd	33	c7	c7	«Êí¯.aQ.Nê.‡Í3ÇÇ
00000040	71	31	30	9c	c4	28	0e	b4	24	3d	11	54	f0	44	f9	cf	q10œÄ(.´$=.TðDùÏ
00000050	62	96	d9	bf	f7	39	7e	43	90	98	7f	e6	32	03	da	0d	b–Ù¿÷9~C...æ2.Ú.
00000060	e4	02	78	b3	a5	4f	5d	dc	69	75	fa	04	f7	49	84	9e	ä.x³¥O]Üiuú.÷I„ž
00000070	1a	62	59	5a	9f	63	0b	07	95	91	3d	e0	15	3e	3a	ac	.bYZŸc..•'=à.>:¬
00000080	38	8c	45	fb	9d	85	0c	fe	91	35	41	d6	c0	83	98	f2	8ŒEû.…..þ'5AÖÀƒ.ò
00000090	c8	83	32	a8	2f	df	00	28	1d	62	fc	dc	4f	e7	e4	6a	Èƒ2¨/ß.(.büÜOçäj
000000a0	e9	0c	51	c5	c8	06	b4	11	64	e3	3a	b9	2c	96	86	2e	é.QÅÈ.´.dã:¹,–†.
000000b0	06	8b	0c	16	c0	99	90	b8	38	1a	00	da	79	15	b6	7f	.‹..À™.¸8..Úy.¶.
000000c0	e4	a2	0f	59	9b	0f	1b	6d	48	19	13	c7	b9	53	8c	ee	ä¢.Y›..mH..Ç¹SŒî
000000d0	63	91	44	f4	15	61	ba	92	e4	fe	75	1d	1e	24	2c	d8	c'Dô.a°'äþu..$,Ø
000000e0	8f	51	d6	95	51	98	87	13	6a	7c	15	aa	bd	7b	40	04	.QÖ•Q.‡.j|.ª½{@.
000000f0	49	22	01	41	30	a9	1f	17	0f	66	cc	b3	c1	39	46	3a	I".A0©...fÌ³Á9F:
00000100	7e	90	9a	37	aa	86	3f	b2	78	05	fc	97	31	c0	9c	8c	~.š7ª†?²x.ü—1ÀœŒ
00000110	79	06	7e	79	93	0a	40	65	46	b2	4c	9a	62	9b	26	c2	y.~y".@eF²Lšb›&Â
00000120	ce	2a	4b	e4	8f	58	9a	37	5f	eb	73	1f	c4	ab	22	5c	Î*Kä.Xš7_ës.Ä«"\
00000130	11	84	8c	f8	9e	29	1f	b2	71	33	97	0c	06	36	18	47	.„Œøž).²q3—..6.G
00000140	4a	89	28	01	ed	d6	8f	54	69	8c	5e	5b	50	67	46	f6	J‰(.íÖ.TiŒ^[PgFö
00000150	76	5a	6f	7f	12	25	de	a4	da	11	40	fe	b6	0f	65	07	vZo..%Þ¤Ú.@þ¶.e.
00000160	45	24	1c	69	36	95	88	3d	cb	21	e6	fb	fe	fb	b8	5a	E$.i6•ˆ=Ë!æûþû¸Z
00000170	29	91	94	80	36	a5	2b	5d									)'"€6¥+]


010 Editor - D:\HTTPD\home\homm\www\test_caves_emty.h3m.010
File Edit Search View Format Scripts Templates Tools Window Help
Workspace
Open Files
D:\HTTPD\home\homm\...\010\heroes3\hstring.bt
D:\HTTPD\home\...\test_caves_emty.h3m.010
Favorite Files
Recent Files
D:\HTTPD\home\...\010\H3M_SoD_Template.bt
D:\HTTPD\home\homm\...\test_caves_emty.h3m
D:\HTTPD\home\...\test_caves_emty.h3m
D:\HTTPD\home\h3-front-end\maps\big_test.h3m
D:\HTTPD\home\...\A_Viking_We_Shall_Go.h3m
D:\HTTPD\home\...\maps\big_test.h3m
D:\HTTPD\home\...\maps\big_test.h3m.010
D:\HTTPD\home\...\maps\test_caves_emty.h3m.010
Bookmarked Files
Files Explorer
Inspector
Type Value
Signed Byte 0
Unsigned Byte 0
Signed Short 0
Unsigned Short 0
Signed Int 167772160
Unsigned Int 167772160
Signed Int64 55169095603060736
Unsigned Int64 55169095603060736
Float 6.162976e-33
Double 5.6961892475381e-305
Half Float 0
String
Unicode
DOSDATE
DOSTIME 00:00:00
Auto Variables Bookmarks Func
Startup
test_caves_emty.h3m.010
Edit As: Hex
Run Script
Run Template: H3M_SoD_Template.bt
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0040h: 01 00 00 12 06 00 00 03 FF 00 00 00 00 00 02 00 ....ÿ.......
0050h: 00 00 03 00 00 00 00 05 00 00 00 00 00 00 00 28 ....(
0060h: 00 00 F0 00 00 FF 00 00 00 00 00 00 00 00 6D 00 ..ð..ÿ........m.
0070h: 00 5C 00 00 FF 00 00 00 00 00 00 00 00 33 00 00 .\..ÿ........3..
0080h: 00 00 00 FF 00 00 00 00 00 00 00 00 60 00 00 00 ...ÿ........`...
0090h: 00 00 FF 00 00 00 00 00 00 00 00 01 00 00 68 00 ..ÿ..........h.
00A0h: 00 FF 00 00 00 00 00 00 00 00 D3 00 00 00 00 00 .ÿ........Ó.....
00B0h: FF 00 00 00 00 00 00 00 00 0A 00 00 C4 00 00 FF ÿ.....Ä..ÿ
00C0h: 00 00 00 00 00 FF FF 00 C7 FF FF FF FF FF FF FF .....ÿÿ.Çÿÿÿÿÿÿÿ
00D0h: FF FF FF FF FF FF FF FF FF FF FF 0F 00 00 00 00 ÿÿÿÿÿÿÿÿÿÿÿ.....
00E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0100h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0110h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 ....
0120h: 00 00 00 06 00 00 00 F1 EB F3 F5 20 31 0D 00 00 ....ñëóõ 1...
struct FreeHeroes fh
uchar free_heroes[20]
uchar free_heroes[14] = 255
Template Results - H3M_SoD_Template.bt
Name Value Start Size Color Comment
struct PLAYER_ATTRIBUTES purple 98h Fh Fg: Bg:
struct PLAYER_ATTRIBUTES teal A7h Fh Fg: Bg:
struct PLAYER_ATTRIBUTES pink B6h Fh Fg: Bg:
char isHuman 0 B6h 1h Fg: Bg:
char isComputer 0 B7h 1h Fg: Bg:
char behavior 0 B8h 1h Fg: Bg:
char isCityTypesOpt 10 B9h 1h Fg: Bg:
short cityTypes 00000000 00000000,b BAh 2h Fg: Bg:
char randomCity -60 BCh 1h Fg: Bg:
char mainCity 0 BDh 1h Fg: Bg:
uchar random_hero 0 BEh 1h Fg: Bg:
uchar hero_type 255 BFh 1h Fg: Bg:
char junk 0 C0h 1h Fg: Bg:
uint heroes_count 0 C1h 4h Fg: Bg:
struct SpecialVictoryConditions svc C5h 1h Fg: Bg:
struct SpecialLossConditions slc C6h 1h Fg: Bg:
struct Teams cmd C7h 1h Fg: Bg:
struct FreeHeroes fh C8h 38h Fg: Bg:
struct Artefacts arts 100h 12h Fg: Bg:
Output
Executing template 'D:\HTTPD\home\homm\www\010\H3M_SoD_Template.bt' on 'D:\HTTPD\home\homm\www\test_caves_emty.h3m.010'...
*WARNING Line 732: Variable 'gevents' not generated since array size is zero.

Я решил использовать homm3tools...

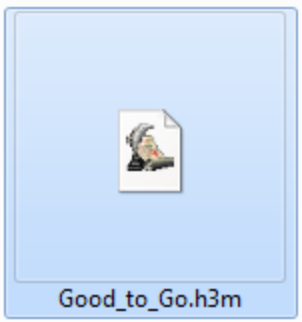
Это набор библиотек и утилит, для работы с картами

1. Парсер разных форматов
2. Генератор карт
3. Рендер надписей из деревьев
4. И, даже, игра "змейка"

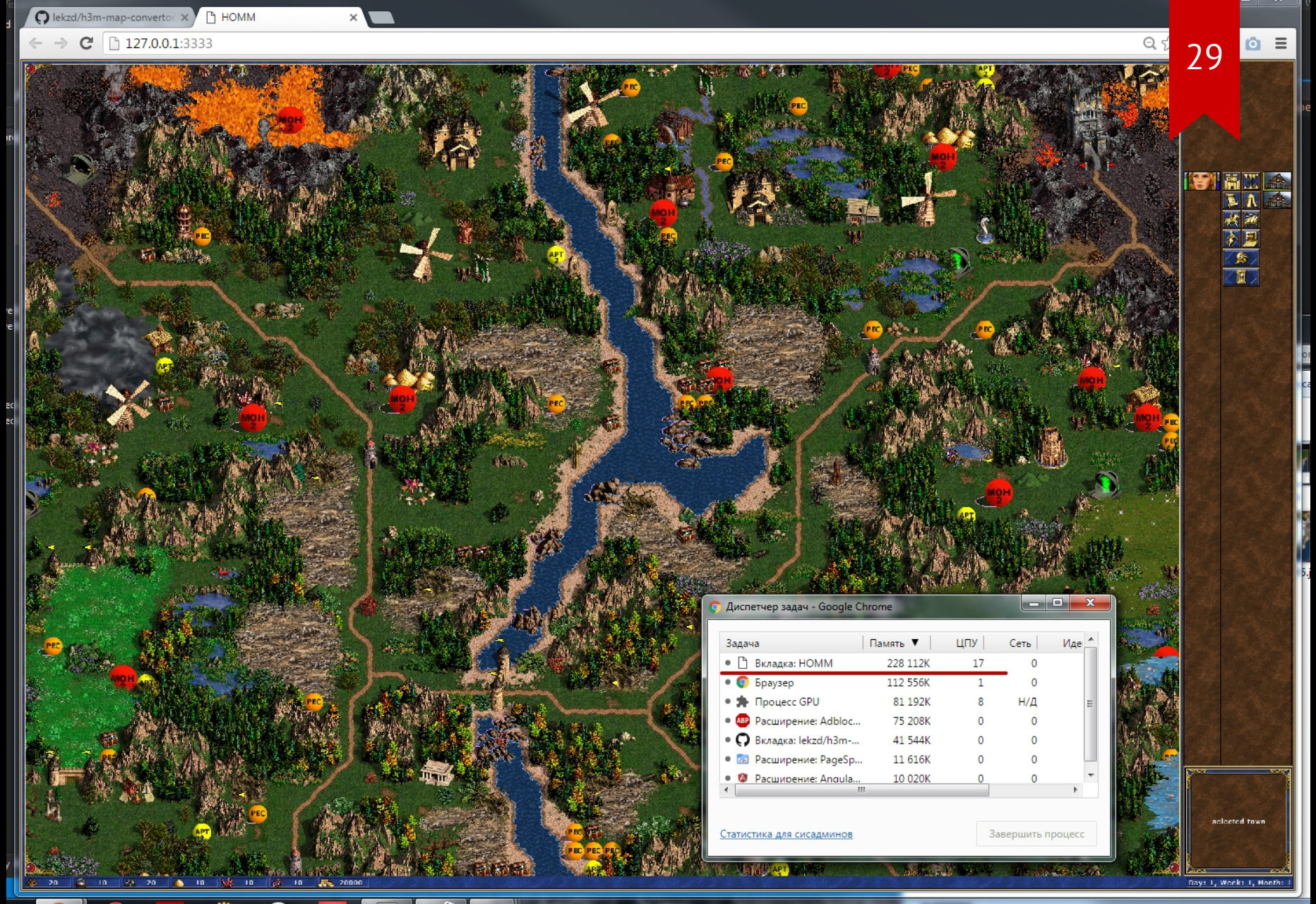


...и сделал свой конвертор

h3m-map-convertor



Данных становилось все больше, кол-во объектов росло...


lekzd/h3m-map-convertor
HOMM
127.0.0.1:3333
Диспетчер задач - Google Chrome
Задача
Память
ЦПУ
Сеть
Вкладка: HOMM 228 112K 17 0
Браузер 112 556K 1 0
Процесс GPU 81 192K 8 Н/Д
Расширение: Adbloc... 75 208K 0 0
Вкладка: lekzd/h3m-... 41 544K 0 0
Расширение: PageSp... 11 616K 0 0
Расширение: Angula... 10 020K 0 0
Статистика для сисадминов
Завершить процесс

Все тормозит!

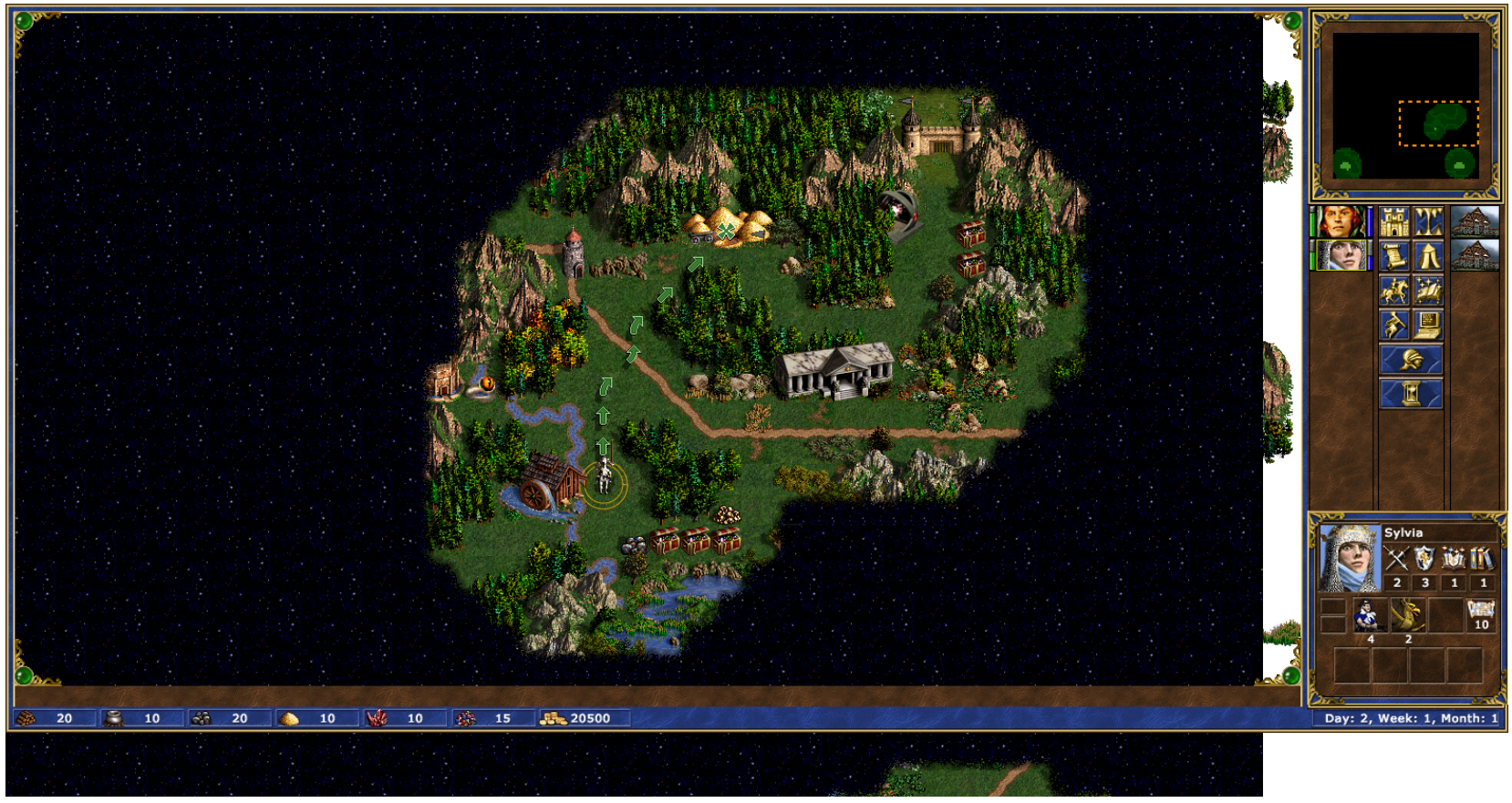
#1 Отрисовка карты

Canvas это не DOM

```
01. const ctx = canvas.getContext('2d')
02.
03. ctx.drawImage(hero, 0, 0)
04. ctx.clearRect(0, 0, 100, 100)
05. ctx.drawImage(hero, 100, 0)
06. ctx.clearRect(0, 0, 100, 100)
07. ctx.drawImage(hero, 200, 0)
```

А если под героем трава...

```
01. const ctx = canvas.getContext('2d')
02.
03. ctx.drawImage(hero, 0, 0)
04. ctx.drawImage(grass, 0, 0)
05. ctx.drawImage(hero, 100, 0)
06. ctx.drawImage(grass, 100, 0)
07. ctx.drawImage(hero, 200, 0)
```


Sylvia
2
3
1
1
10
4
2
20
10
20
10
10
15
20500
Day: 2, Week: 1, Month: 1

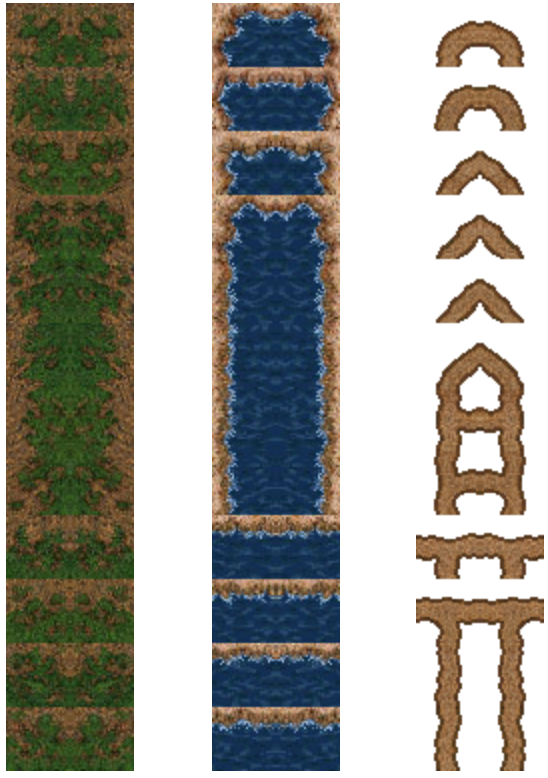
Я просто использую 3 <canvas>

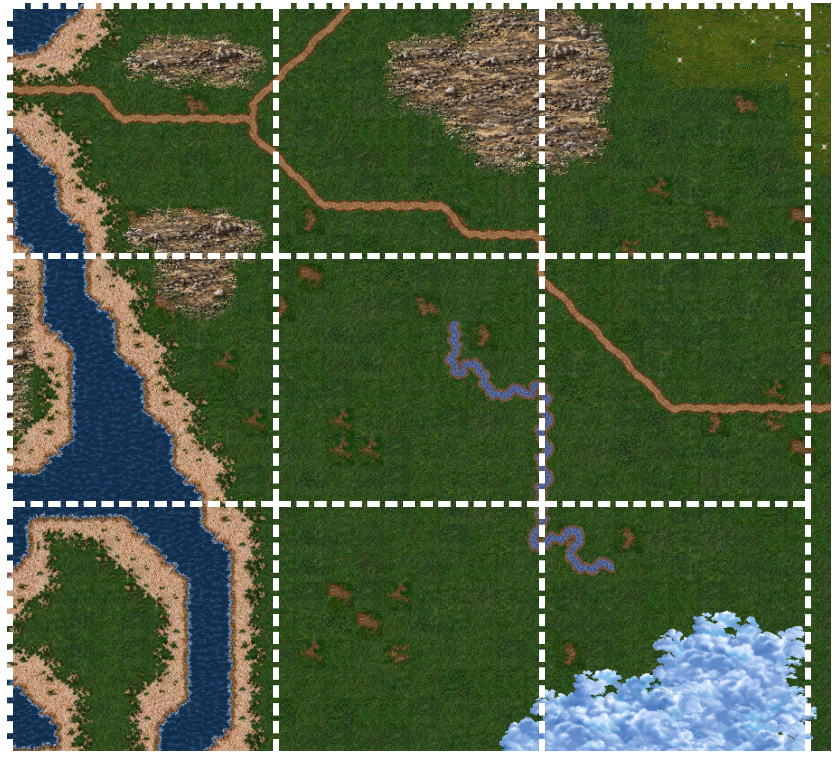
```
01. <canvas id="terrain">
02. <canvas id="objects">
03. <canvas id="ui">
```

Алгоритм рисования террейна

1. Взять тайл типа почвы
2. Нарисовать его со смещением и поворотом
3. Наложить реки
4. Наложить дороги
5. И еще остались особые типы почв…

Рисуем все сразу и кладем в кэш





Как плавно двигать карту

1. Сдвигаем карту при помощи

   transform: translate()

2. Каждые 32 пикселя

   увеличиваем left на 32px

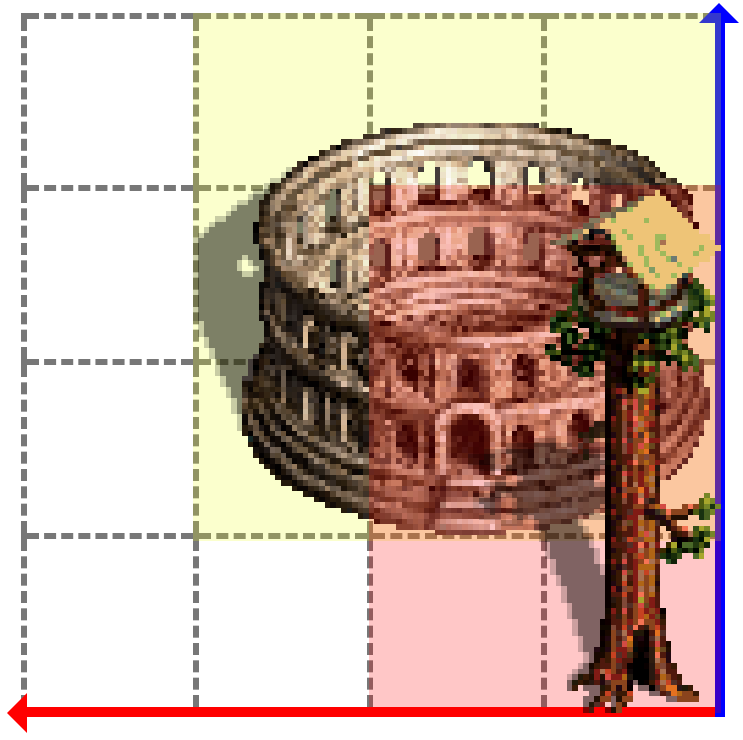
   рисуем еще одну полоску

Иллюзия непрерывного блока

спасибо Яндекс-картам за идею

#2 Отрисовка объектов

Особенности рисования объектов



Алгоритм рисования объектов

1. Сортируем массив объектов по Y нижней границы
2. Фильтруем те, что не попадают в окно
3. ...тут масса проверок...
4. Рисуем текстуру объекта
5. Если нужно рисуем флаг игрока

А ведь кол-во объектов на карте может достигать over 9000!

Построим дерево!

1. Каждая ветвь - Y объекта
2. Объекты на ветви отсортированы по X

+ отсечение видимых объектов
+ корректное перекрытие
+ более дешевая итерация

```
01. const renderTree = {
02.     32: [object, object, ...]
03.     64: [object, object]
04.     96: [object, object, ...]
05.     128: [object]
06. }
```

Если заглянуть в функцию рисования объекта...

```
01. const object = getObject(id)
02. const {x, y} = getAnimationFrame(object)
03. const offsetleft = getMapLeft()
04. const offsetTop = getMapTop()
05.
06. context.drawImage(object.texture, x - offsetleft, ...
```

То, все рисование сводится к:

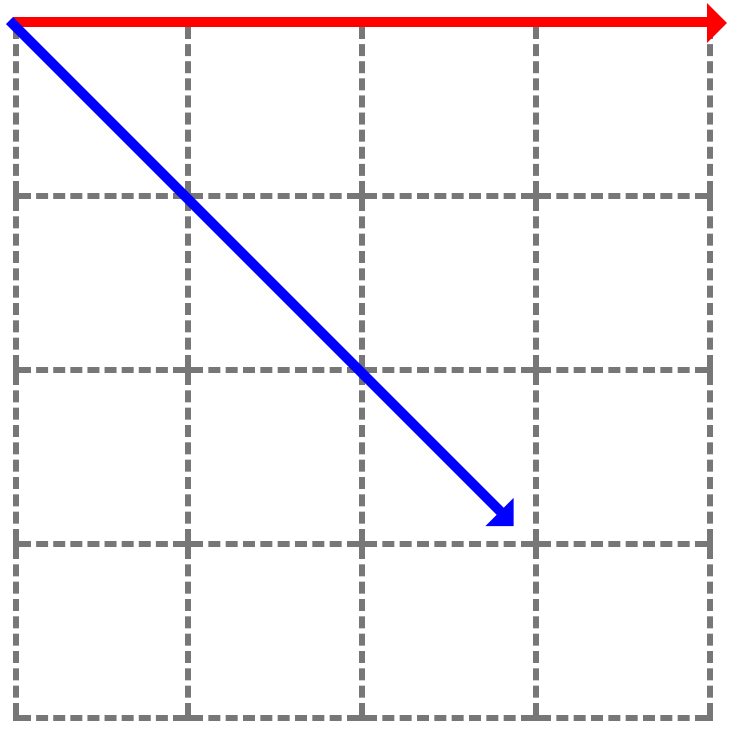
```
context.drawImage(object.texture, x, y)
```

Поэтому мы просто возьмем и запишем их в Render tree!

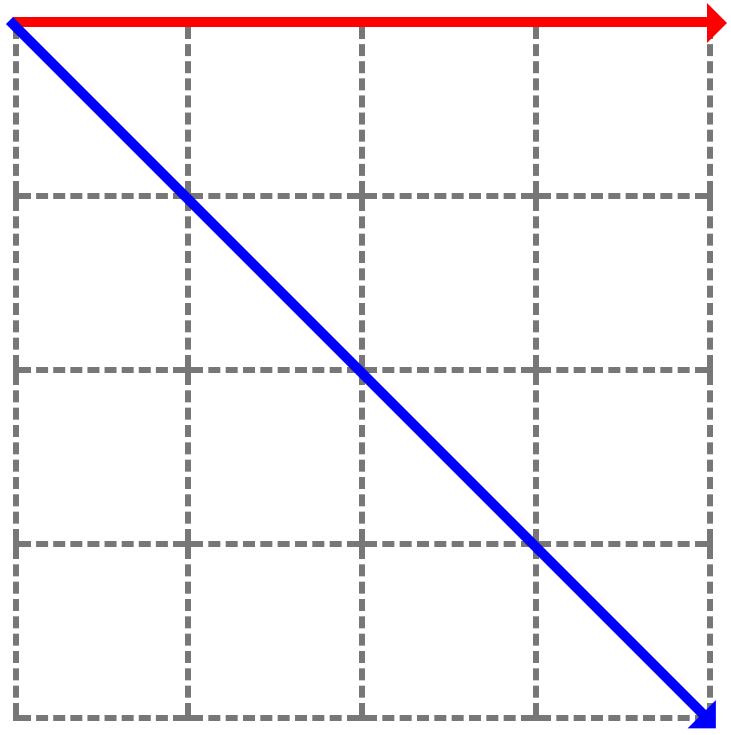
```
01. const drawer = context.drawImage.bind(context, ...)
02. renderTree.add(x, y, drawer)
```

И получим отличный прирост производительности!

Минутка геометрии



Минутка **герое** метрии 🏇



Чтобы достичь одинаковой скорости...

4 шага

~6 шагов

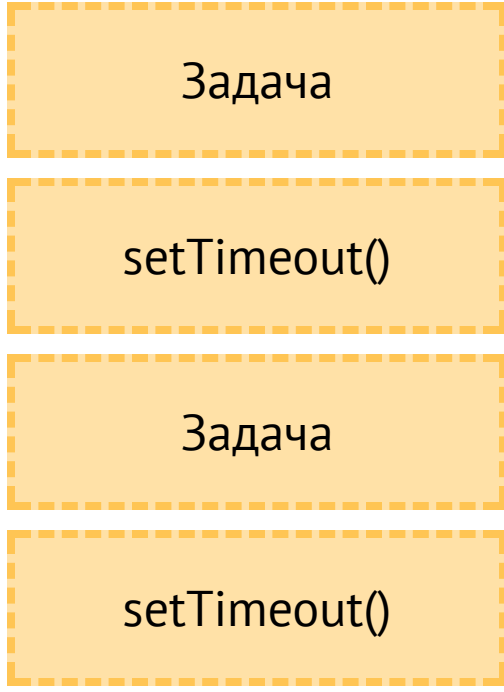
`setTimeout` (() => setTimeout())

vs

`setInterval` ()

vs

`requestAnimationFrame` ()



Promise()

Подробнее об этом в статье Джейка Арчибальда

Возьмем все и сразу

```
01. new Promise(resolve => {
02.     setTimeout(() => {
03.         // расчеты для анимации
04.         requestAnimationFrame(() => /* рисование */)
05.         resolve()
06.     })
07. })
```

Построим из этого последовательность

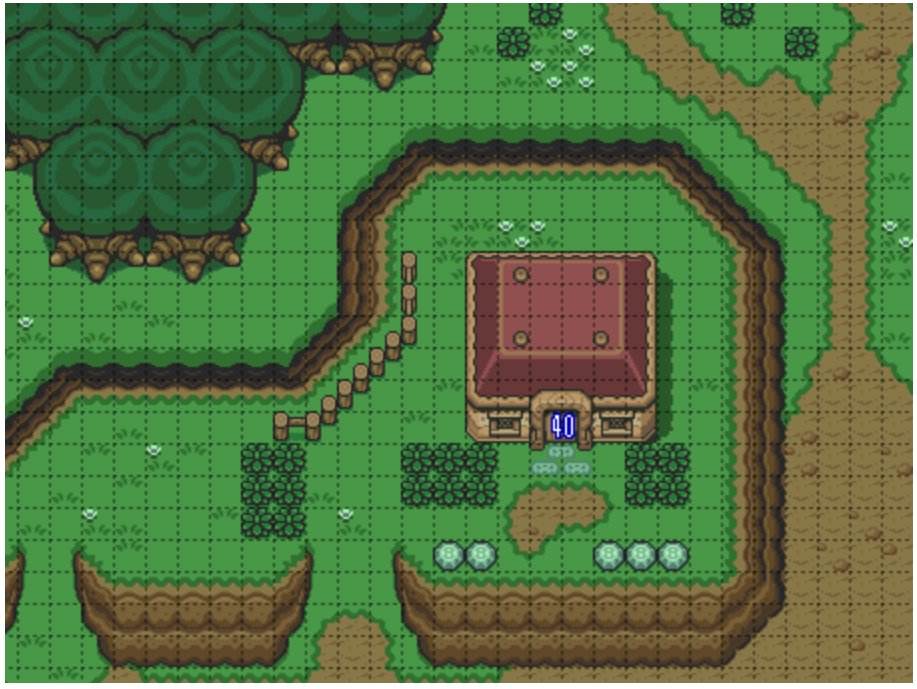
```
01. startAnimation()
02.     .then(step)
03.     .then(step)
04.     .then(step)
05.     .then(step)
06.     .then(doAction)
07.     .then(endAnimation)
```

А лучше сразу декларативную абстракцию

```
AsyncSequence([
    startAnimation, [
        step
        step
        ...],
    doAction,
    endAnimation
])
```

#3 Хранение данных

Вся карта — это сетка



* а точнее — тайлы

Каждый тайл это

1. Тип (вода, земля, дерево)
2. Проходимость/стоимость перемещения
3. Наличие события
4. Флаг "Кем занят"
5. ...

Представим это в коде

```
01. const map = [
02.     [{...}, {...}, {...}, {...}, {...}, {...}],
03.     [{...}, {...}, {...}, {...}, {...}, {...}],
04.     [{...}, {...}, {...}, {...}, {...}, {...}],
05.     ...
06. ]
07. const tile = map[1][3]
```


282/1716

Чтобы получить свойство тайла

1. Запросить массив тайлов
2. Запросить массив массива для строки
3. Запросить объект тайла
4. Запросить свойство объекта

Хранить все в одном месте — это медленно

```
01. const tile = {
02.     // данные для отрисовки
03.     render: {...},
04.     // данные для поиска пути
05.     passability: {...},
06.     // данные которые нужны значительно реже
07.     otherStuff: {...},
08. }
```

Быстрее всего читать данные из массива

[1,0,1,1,0,0,1,1]

Если заранее делать необходимые расчеты

tiles [{…} , {…}, {…}]

+

objects [{…} , {…}, {…}]

=

[1 , 0, 2]

Делаем разные массивы

Цикл отрисовки	массив функций отрисовки
Поиск пути	массив чисел
Ассоциация объектов к тайлам	массив строк
Дополнительные свойства тайлов	массив чисел
Для игровой логики	Map объектов с их ID

Остается только своевременное обновление данных из медленных хранилищ в быстрые

Как можно уйти от массивов массивов

```
01. const map = [
02.     [{...}, {...}, {...}, {...}, {...}, {...}],
03.     [{...}, {...}, {...}, {...}, {...}, {...}],
04.     [{...}, {...}, {...}, {...}, {...}, {...}],
05.     ...
06. ]
```

Развернем массив, получаем прирост ~50%

```
01. const map = [{...}, {...}, {...}, {...}, ...]
02.
03. const tile = map[y * width + x]
```

Test		Ops/sec
Classic	`var v = classic[1][1];` `classic[2][2] = 255;`	421,908,234 ±1.26% 47% slower
Flat	`var v = flat[1 * width + 1];` `flat[2 * width + 2] = 255;`	790,122,673 ±1.05% fastest

Пробуем ускорить и дальше

```
01. const map = [{...}, {...}, {...}, {...}, ...]
02.
03. const tile = map[y * width + x]
04. map.forEach((value, index) => {
05.     const y = Math.floor(index / width)
06.     const x = index - (y * width)
07. })
```

Power of 2!

$2^n === 2 << n$

$Math.floor(X / 2^n) === X >> n$

Test		Ops/sec
Right shift	`x = y >> 1;` `y = x >> 1;`	787,431,451 ±1.46% fastest
Division	`x = y / 2;` `y = x / 2;`	302,518,333 ±1.72% 62% slower

Погружаемся в побитовые сдвиги

```
01. const map = [{...}, {...}, {...}, {...}, ...]
02. const powerOfTwo = Math.ceil(Math.log2(width))
03.
04. const tile = map[y << powerOfTwo + x]
05. map.forEach((value, index) => {
06.     const y = index >> powerOfTwo
07.     const x = index - (y << powerOfTwo)
08. })
```

Степень двойки в видеокарте

Так получился Grid

```
01. const grid = new Grid(32)
02.
03. const tile = grid.get(x, y)
04. grid.forEach((value, x, y) => {})
```

- только квадратные сетки
- неэффективен для сеток стороной более чем 256

#3 UI на Canvas

В начале я создавал UI-элементы так

```
01. const okButton = new Buttton(0, 10, 'Ok')
02. okButton.addEventListener('click', () => { ... })
03. const cancellButton = new Buttton(0, 10, 'Cancel')
04. cancellButton.addEventListener('click', () => { ... })
```

Кол-во параметров росло и росло...

```
01. const okButton = new Buttton({
02.     left: 0,
03.     top: 10,
04.     onClick: () => { ... }
05. })
06. const cancellButton = new Buttton({...})
```

Их становилоь все больше, появился JSON...

```
[
    {
        id: 'okButton',
        options: {
            left: 0,
            top: 10,
            onClick: () => { ... }
        },
    },
```

Но далее я вспомнил, что есть XML

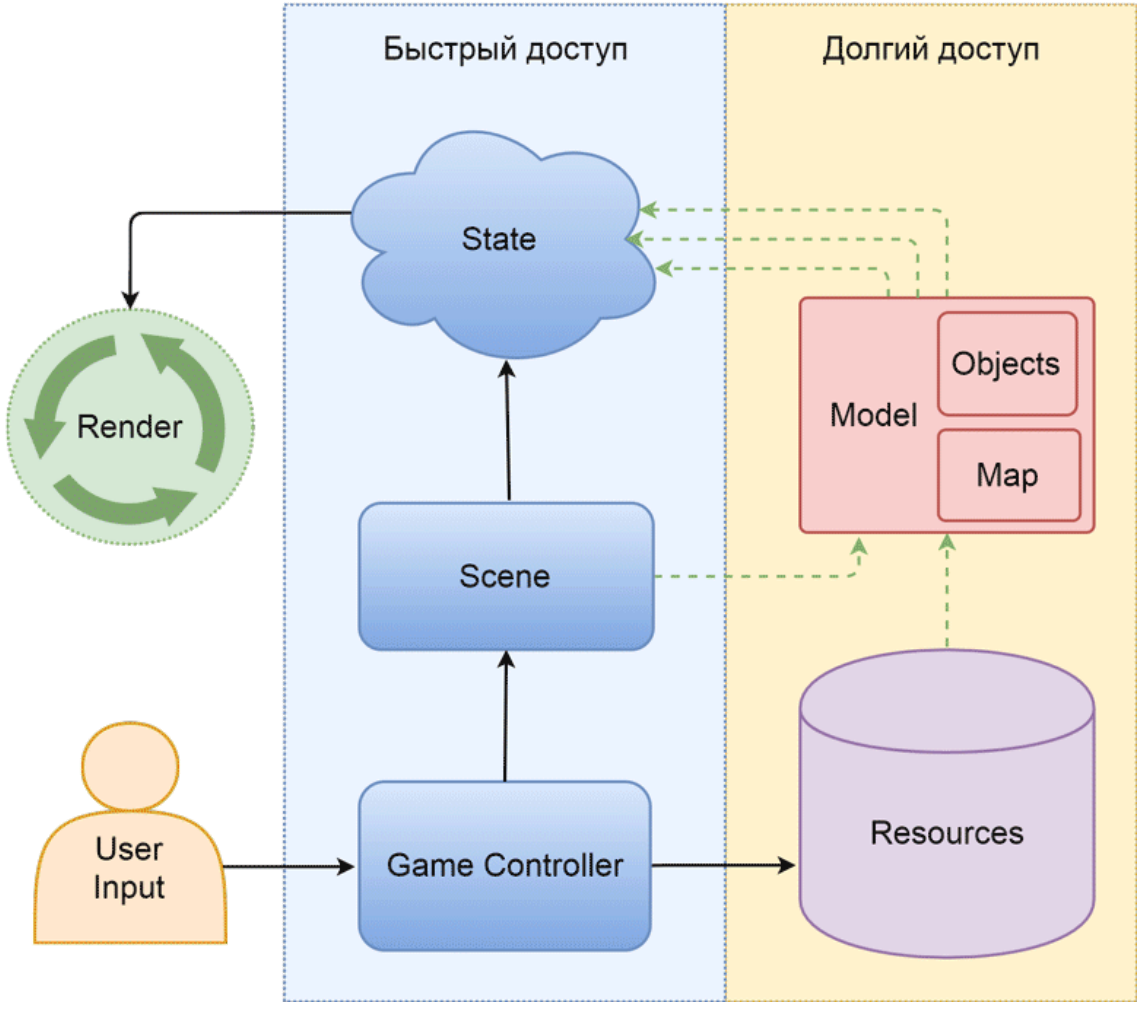
```
01. <button id="okButton"
02.         left="0"
03.         top="10"
04.         onClick="{doSomething()}"
05. />
```

При сборке он собирается в JSON с предыдущего слайда

```
<group id="main" ... >
    <group id="header" ... >
        <text-block ... />
        <button ... />
    </group>
    <group id="footer" ... >
        <any-component ... />
        <button ... />
    </group>
</group>
```

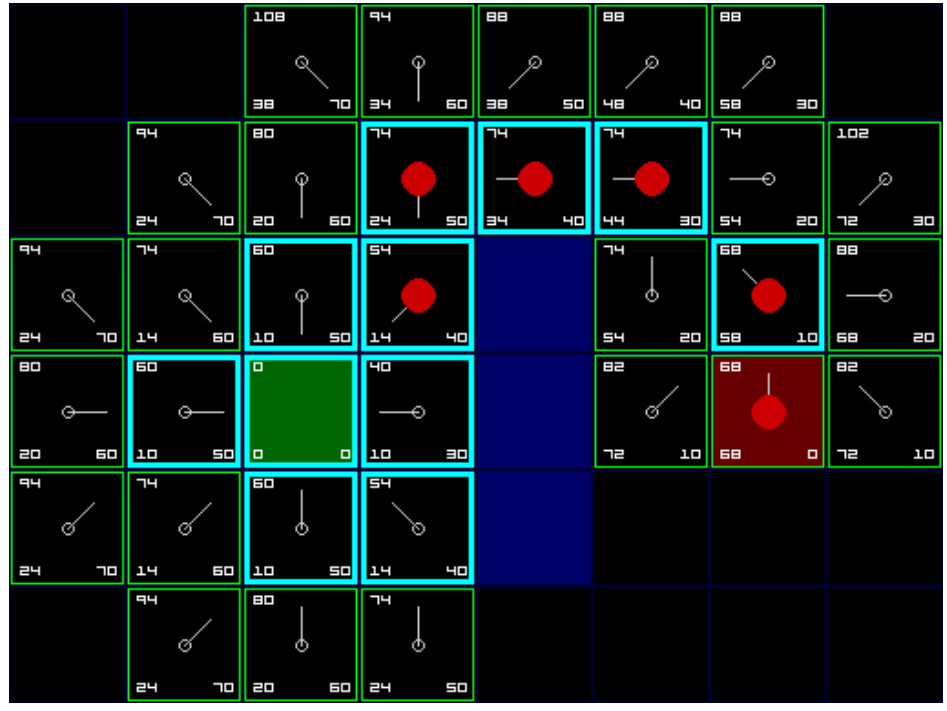
Прямо как в react-canvas

Как это все работает


Быстрый доступ
Долгий доступ
State
Render
Model
Objects
Map
Scene
User
Input
Game Controller
Resources

Например, сбор ресурсов

#1 Поиск пути — алгоритм A*



#2 Анимация движения

После каждого шага нужно обновлять позицию героя в дереве отрисовки

#3 Проверка в конечной точке

1. Делаем запрос к карте и получаем ID объектов в этой точке
2. Они отсортированы как: действия, непроходимые и проходимые
3. Берем первый объект по ID
4. Проверяем можно ли заходить на объект для активации действия

#4 Действие с объектом

```
01. const objectInAction = Objects.get(ID)
02. const hero = Player.activeHero
03. objectInAction.events.dispatch('action', hero)
04. ...
05. this.events.on('action', hero => {
06.     hero.owner.resources.set('gems', this.value)
07.     this.remove()
08. })
```

#5 Удаление объекта

1. Удаляем отрисовку из рендера
2. Удаляем из массивов для поиска пути
3. Удаляем из ассоциативного массива с коодинатами
4. Удаляем обработчики событий
5. Удаляем из массива объектов
6. Обновляем мини-карту, уже без этого объекта
7. Рассылаем событие об удалении этого объекта из текущего стейта

Как обновлять все эти массивы быстрее?

Доля динамических объектов около 10% от всех.

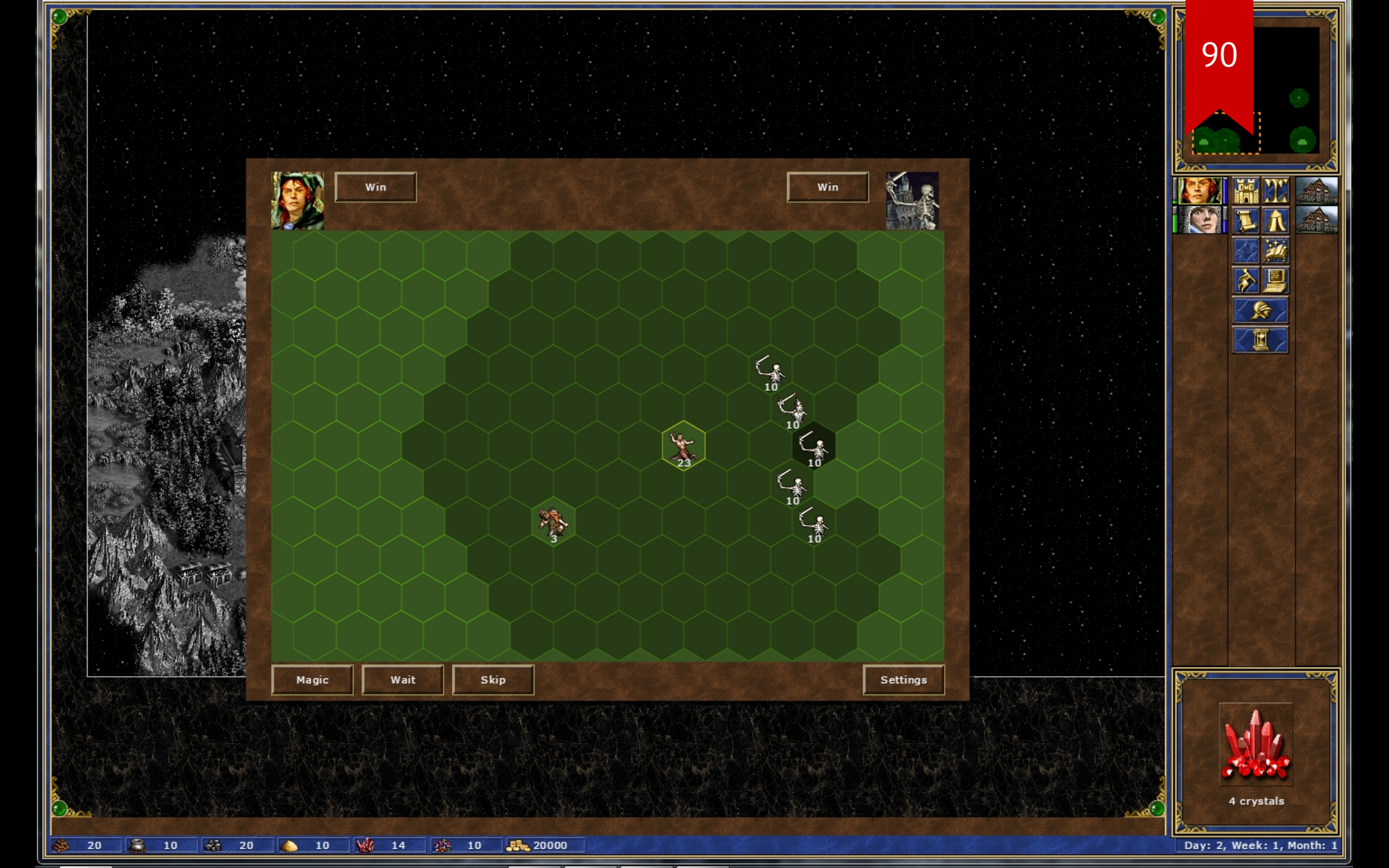
Так почему бы не сэкономить на расчетах:

```
01. // только при загрузке карты, содержит много данных
02. const baseGrid = new Grid(mapSize)
03. // обновляется намного чаще, потому содержит мало данных
04. const dynamicGrid = new Grid(mapSize)
```

Как устроены объекты

```
01. // Объект содержит гарнизон и может быть атакован
02. @Mixin(Attacable)
03. class TownObject extends OwnershipObject {...}
04. // Содержит все для отрисовки флажка, его смены и т.п.
05. class OwnershipObject extends MapObject {...}
06. // Содержит все базовые поля для объекта карты
07. class MapObject {...}
```

Выводы


90
Win
Win
10
10
10
10
10
23
3
Magic
Wait
Skip
Settings
4 crystals
20
10
20
10
14
10
20000
Day: 2, Week: 1, Month: 1


BASEL

Что мне это дало

1. саморазвитие
2. выходы за рамки привычных рабочих задач
3. расширение кругозора
4. знакомство с фанатиками

Зачем делать игры

1. куда интереснее чем сайтики
2. это алгоритмы
3. это красиво

Степень мастерства прямо пропорциональна времени, которое можно потратить на работу вглубь

Полезные ссылки

Про побитовые операторы на JS

Книга Game Programming Patterns из которой я многое вынес

Интерактивная визуализация алгоритмов поиска пути в сетке

FizMig. Та самая спецификация по всем игровым механикам Героев

Демка Героев 3 в браузере

http://homm.lekzd.ru/

Вопросы