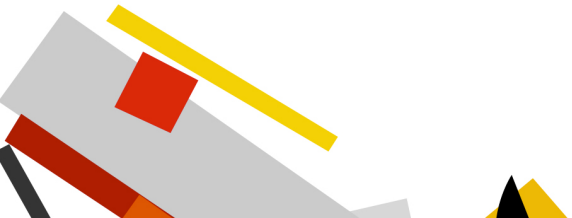
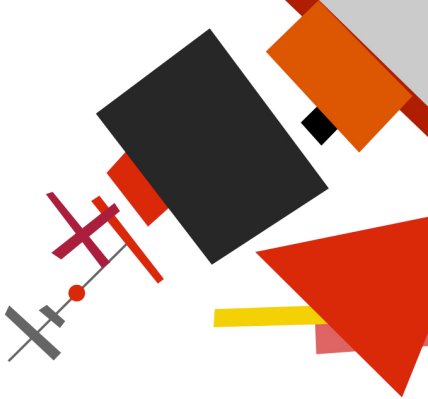
Быстрое прототипирование бэкенда игры с геолокацией на OpenResty, Redis и Docker

Александр Гладыш
CTO, LogicEditor





Профессиональная конференция разработчиков высоконагруженных систем

План доклада

1. Кейс
2. Задача
3. Философия
4. План разработки
5. Docker
6. HTTP API
7. Устройство игрового мира
8. Демонстрация
9. Клиент
10. Итоги
11. Вопросы?

Обо мне

- В разработке ПО с 2002-го,
- большую часть этого времени — в геймдеве (разработка, проектирование, управление),
- вне геймдева — нагруженные интернет-решения, enterprise ПО и др.
- Организатор meetup.com/Lua-in-Moscow

Мобильные игры с геолокацией

Цели прототипирования в общем

- Проверить ряд подходов к построению игрового процесса,
- выработать новые идеи,
- найти, в чём фан, а в чём — нет.

Цели технологической части прототипирования

- С минимальными затратами получить код,
- дающий возможность быстро итерироваться по вариантам геймплея.
- Прояснить на практике технические ограничения жанра.

Результаты

За два календарных месяца (менее 100 человеко-часов) разработан прототип серверной части игры с геолокацией и рудиментарный клиент для неё.

Ведётся быстрое итерирование по вариантам построения игрового процесса и дальнейшая разработка технологической части проекта.

О чём этот доклад?

- Доклад — технологической части проекта,
- не о геймдизайне
- и не о монетизации.

Зачем этот доклад?

Делать игры с геолокацией сейчас проще чем когда-либо.

Я покажу с чего можно начать.

Задача

- Максимально быстро написать сервер для быстрого прототипирования.
- Параллельно с сервером реализовать минимальный клиент.
- Проверять гипотезы уже во время написания, если возможно.
- Выявить основные технические ограничения на проект.

Стадии разработки

- Препродакшен
- Продакшен
- Поддержка

Стадии разработки: Препродакшен

- Препродакшен
  - Поиск геймплея, эскизы, выяснение ограничений.
  - Более глубокая проработка удачных вариантов геймплея.
  - Подготовка к продакшену выбранного варианта.

Продакшен: приоритеты разработки

- Восприятие проекта
- Устойчивость ко взлому
- Масштабируемость
- Производительность
- Стабильность
- Гибкость
- Скорость итерирования
- Простота

Препродакшен: время на вес золота

- Скорость итерирования
- Гибкость
- Простота
- Восприятие проекта
- ~~Стабильность~~
- ~~Производительность~~
- ~~Масштабируемость~~
- ~~Устойчивость ко взлому~~

Фокус на скорость и лёгкость разработки

- Механизмы, а не решения.
- Лучше быстро чем правильно.
- Много коротких итераций.

Лучше быстро чем правильно. Переориентация перфекционизма

- Чем меньше кода тем лучше.
- Плохой код лучше сложного.
- Хаки в механизмах допустимы, хаки в решениях — нет.
- Рефакторить нужно только то, что болит.

Основные этапы разработки

- Выбор геймплея пилотного прототипа.
- Выбор технологического стека.
- Реализация минимального пилотного прототипа.
- Итерирование с гейм-дизайнерами.

Выбор геймплея пилотного прототипа: Задачи

- Максимально простой
- но играбельный
- повод реализовать все базовые игровые сущности и механизмы прототипа.

Геймплей первого прототипа

- Игрок, перемещаясь по карте, ищет расставленных на ней мобов;
- найдя моба может попробовать его поймать с заданной вероятностью успеха;
- успешная поимка моба увеличивает счётчик в характеристиках игрока;
- пойманный моб исчезает с карты;
- через некоторое время моб респавнится в том же месте;
- администраторы могут добавлять новых мобов на карту.

Критерии выбора технологического стека

- Что-то знакомое, на чём можно написать быстро,
- или что-то интересное и зажигающее.
- Главное не забыть вовремя выбросить весь код.

Технологический стек

- Сервер:
  - Redis,
  - OpenResty,
  - Docker.
- Клиент:
  - Одностраничное веб-приложение в браузере,
  - HTML5.

Почему не что-то готовое?

Not Invented Here Syndrome?

И да и нет.

Redis

- Надёжное, хорошо зарекомендовавшее себя решение.
- Работа с координатами из коробки:
  - `GEOADD key longitude latitude member`
  - `GEORADIUS key longitude latitude radius m`
- Достаточно удобный набор примитивов для хранения игровых объектов.
- Хранимые процедуры на Lua.

OpenResty

- Дистрибутив nginx с поддержкой Lua, Redis и многим другим из коробки.
- Очень быстро работает, достаточно дружелюбен, хорошо поддерживается.
- Пригоден как для быстрого прототипирования, так и для продакшена.

Docker

- Воспроизводимая кроссплатформенная среда для разработки.
- Хорошо снимает боль по настройке окружения разработчика.
- Окружение разработчика можно быстро превратить в прототип серверного окружения.
- Требует обновления до достаточно свежей версии.

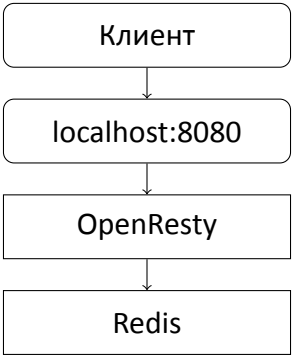
Браузер, HTML5

- На начальном этапе сервер важнее.
- Бои в augmented reality и прочие рюшечки делать не нужно, можно представлять в голове.
- На HTML5 можно быстро написать дёшевый и сердитый клиент.
- Есть ограниченный (но достаточный) доступ к данным геолокации.

Docker: как установить на Ubuntu

- В Ubuntu, традиционно, старая версия.
- За `wget | sh` больно бьём по рукам.
- `docker` и `docker-machine` устанавливаем из apt-репозитория докера.
- `docker-compose` устанавливаем через `pip install`.
- Про установку на других платформах читаем официальную документацию.

Docker на машине разработчика



docker-compose.yml для разработки: Redis

```
version: "2"
services:
  redis:
    image: redis
    volumes:
      - ./redis:/data
    command: redis-server --appendonly yes
  openresty: <...>
```

OpenResty: интересные места nginx.conf (в сокращении)

```
error_log logs/error.log notice;
http {
  include resolvers.conf;
  lua_package_path "$prefix/lualib/?.lua;;";
  lua_code_cache off; # TODO: Enable on production!
  server {
    listen 8080;
    include mime.types;
    default_type application/json;
    location / { index index.html; root static/; }
    location = /api/v1/ { content_by_lua_file 'api/index.lua'; }
  }
}
```

OpenResty: Dockerfile

```
FROM openresty/openresty
COPY bin/entrypoint.sh /usr/local/bin/openresty-entrypoint.sh
COPY nginx/conf /usr/local/openresty/nginx/conf
COPY nginx/lualib /usr/loca/openresty/nginx/lualib
COPY nginx/lua /usr/loca/openresty/nginx/lua
COPY nginx/static /usr/loca/openresty/nginx/static
ENTRYPOINT /usr/local/bin/openresty-entrypoint.sh
```

OpenResty: entrypoint.sh

```
#!/bin/sh
grep nameserver /etc/resolv.conf \
  | awk '{print  "resolver " $2 ";"}' \
  > /usr/local/openresty/nginx/conf/resolvers.conf
/usr/local/openresty/bin/openresty -g 'daemon off;' "$@"
```

docker-compose.yml для разработки: OpenResty

```
<...>
openresty:
  build: .
  ports:
    - "8080:8080"
  volumes:
    - ./nginx/lualib:/usr/local/openresty/nginx/lualib:ro
    - ./nginx/api:/usr/local/openresty/nginx/api:ro
    - ./nginx/static:/usr/local/openresty/nginx/static:ro
  links:
    - redis
```

Вызовы API

- Пользовательские вызовы:
  - `/` состояние игрового мира,
  - `/go/:go-id/` состояние игрового объекта,
  - `/go/:go-id/act/:action-id` выполнение действия.
- Системные вызовы:
  - `/register` создание пользователя,
  - `/reset` сброс базы в исходное состояние,
  - `/patch` апгрейд базы до текущей версии.
- NB: Админку (бэкофис) не делаем, используем внутриигровые механики для администрирования игрового мира.

Игровой объект

- С точки зрения сервера игровой мир состоит из игровых объектов.
- Игровой объект имеет численные характеристики и действия.
- Игровые объекты могут иметь координаты.
- Игровые объекты без координат должны принадлежать другим объектам или быть их прототипами.

Цепочка прототипов

- Игровой объект может иметь прототип.
- Игровой объект — прототип в свою очередь также может иметь прототип.
- Игровой объект наследует характеристики и действия своих прототипов.

Характеристики

- Характеристика — именованное численное свойство игрового объекта.
- Если у игрового объекта нет какой-то характеристики, её значение берётся у ближайшего прототипа по цепочке (если не нашли — 0).

Действия

- Действие на игровом объекте — идентификатор из таблицы обработчиков действий.
- Действие может быть инициировано игроком, если у него достаточно на это прав.

Моб: Зелёная Жаба

```
{
  id = 'proto.mob.collectable';
  chrs = { respawn_dt = 10 * 60 };
  actions = { 'mob.collect' };
};
{
  id = 'proto.mob.toad.green';
  proto_id = 'proto.mob.collectable';
  chrs = { escape_chance = 0.25 };
};
{
  id = 'fa2eb7bca46c11e6be447831c1cebc82';
  proto_id = 'proto.mob.toad.green';
  geo = { lat = 55.7558, lon = 37.6173 };
};
```

Действие: поймать моба

```
ACTIONS['mob.collect'] = function(target, initiator)
  if
    math.random() * initiator.chrs.collect_skill >
    target.chrs.escape_chance
  then
    -- Inc number of catches for this mob type
    go_inc_chr(initiator.id, target.proto_id, 1)
    go_schedule_action_initiation( -- Schedule respawn
      target.chrs.respawn_dt, 'mob.spawn',
      { proto_id = target.proto_id, pos = target.pos },
      initiator.id
    )
    go_remove(target.id) -- Mob is caught, remove
  end
end
```

Выполнение отложенных действий

```
local timestamp = os.time()
local action_ids = redis:zrangebyscore('da', '-inf', timestamp)
for i = 1, #action_ids do
  -- Execute action_ids[i] action
end
redis:zremrangebyscore('da', '-inf', timestamp)
```

Игрок

```
{
  id = 'proto.user';
  chrs = { vision = 100, reach = 50 };
};
{
  id = 'user.1';
  geo = { lat = 55.7558, lon = 37.6173 };
  chrs = { collect_skill = 0.5 };
};
```

Предмет: Админская шапка

```
{
  id = 'proto.item.wearable';
  actions = {
    ['item.don'] = { enabled = true };
    ['item.doff'] = { enabled = false };
  };
}
{
  id = 'proto.item.admin-hat';
  proto_id = 'proto.item.wearable';
  grants = { 'user.admin' };
  chrs = { collect_skill = 0.25 };
};
```

Выдадим админскую шапку пользователю

```
local hat = go_new('proto.item.admin-hat')

assert(go_get('user.1').stored[1] == nil)

go_store('user.1', hat.id)

assert(go_get('user.1').stored[1] == hat.id)
```

Хранение (”storage”)

- Игровой объект может ”хранить” другие объекты.
- Хранимые объекты не ”видны” извне хранящего объекта.
- Пользователю доступны действия непосредственно хранимых им объектов.
- Характеристики хранимых объектов никак не влияют на характеристики хранящих их объектов.

Действия на админской шапке

```
ACTIONS['item.don'] = function(target, initiator)
  go_unstore(initiator.id, target.id)
  go_attach(initiator.id, target.id)
  go_disable_action(target.id, 'item.don')
  go_enable_action(target.id, 'item.doff')
end

ACTIONS['item.doff'] = function(target, initiator)
  go_attach(initiator.id, target.id)
  go_store(initiator.id, target.id)
  go_disable_action(target.id, 'item.doff')
  go_enable_action(target.id, 'item.don')
end
```

Прикрепление / надевание (”attachment”)

- К игровому объекту могут быть ”прикреплены” другие объекты.
- Прикреплённые объекты видны извне родительского объекта.
- Пользователю доступны действия прикреплённых непосредственно к нему объектов.
- Характеристики прикреплённых объектов прибавляются к характеристикам родительских объектов.

Предмет: Спавнилка зелёных жаб

```
{
  id = 'proto.item.spawner.toad.green';
  actions = {
    ['mob.spawn'] = {
      requires = { 'user.admin' };
      param = { proto_id = 'proto.mob.toad.green' };
    };
  };
};
```

Права на действия

- Действие доступно для выполнения только если `grants` игрока содержит все записи из `requires` действия.
- Прикреплённые к игроку (”надетые”) предметы добавляют ему свои `grants`.

Демонстрация

- Текущую версию приложения вы можете найти на geo.logiceditor.com.
- Ссылка на репозиторий с кодом будет опубликована там же на этой неделе, следите за анонсами в Twitter @agladysh.
- Самый первый клиент — всегда `curl`. API можно пощупать им, добавив к адресу приложения `/api/v1`.

Как работает клиент?

- Инициализируются геолокация и гуглекарты.
- Текущая позиция отправляется на сервер.
- Сервер возвращает перечень видимых объектов с возможными действиями.
- Объекты помечаются маркерами на карте и выводятся под ней вёрсткой.
- Ожидаем активации действия пользователем либо смены координат.

NB:

- Для генерации имён жаб на основе их идентификаторов используется chance.js.
- Перерисовку лучше проводить по таймеру, вне зависимости от цикла обновления данных.

Геолокация на HTML5

- `navigator.geolocation.watchPosition(callback, options)`.
- В Chrome пользуйтесь панелью разработчика Sensors для отладки геолокации.
- В Chrome по соображением безопасности отключена геолокация для протокола HTTP (за исключением сайтов на localhost). Используйте HTTPS, например, с сертификатами от Let’s Encrypt.

Google Maps

```
new google.maps.Map(assert(document.getElementById('map')), {
  center: new google.maps.LatLng(pos.lat, pos.lon),
  zoom: 18,
  mapTypeId: google.maps.MapTypeId.ROADMAP,
  disableDefaultUI: true,
  disableDoubleClickZoom: true,
  draggable: false,
  scrollwheel: false,
  styles: [ { featureType: "poi",
    stylers: [ { visibility: "off" } ]
  } ]
});
```

Проблемы проекта

- Шумные данные от GPS.
- Крайне низкая точность геолокации в зданиях.
- ...

~~Проблемы~~ Технические ограничения проекта

- Шумные данные от GPS.
- Крайне низкая точность геолокации в зданиях.
- ...

Нет проблем. Есть ограничения, под которые нужно подстраивать геймплей. Часть из них решается технологически. Нужно ли тратить время на это решение — один из вопросов, на которые должен ответить этап препродакшена.

Недостающие механизмы

- Самое крупное — система событий.
- Много мелких функций, например счётчик пройденных метров.

Ошибки

- Поздновато добавили геолокациию.
- Поздновато добавили карту в клиенте.
- Мало выходили на улицу чтобы тестировать.
- ...
- Найдите сами, сравнив код на слайдах с кодом в проекте.

Итоги

- Относительно малыми усилиями
- мы сделали крошечную мобильную игру с геолокацией
- и заложили фундамент для быстрой разработки большого числа несложных прототипов
- для поиска удачных вариантов геймплея в этом жанре.

Благодаря чему разработка быстрая?

- В проекте небольшой объём простого кода,
- использующего базовые механизмы и надёжные сторонние решения
- для решения большого числа поставленных геймдизайнером задач.

- Аккуратное расширение возможностей этого кода
- ещё больше расширит круг задач, которые можно будет решить,
- поменяв несколько строк в конфиге.

Векторы развития проекта

- Новые варианты геймплея
- Новые функции и механизмы
- Решение технических проблем

Как работать с гейм-дизайнером на ранних этапах прототипирования?

- Быстро итерироваться. В идеале — садиться рядом, кодить и сразу получать фидбек. Пробовать самому иногда делать рутинную часть работы гейм-дизайнера, чтобы лучше понять, что именно мешает и тормозит процесс.
- В первую очередь исправлять мешающие тестировать геймплей баги, потом улучшать старые механизмы в коде и добавлять новые.
- Если для новой геймплейной фичи нет механизма, добавлять его, хотя бы в самой грубой форме, а не реализовывать решение в лоб.
- Все прочие задуманные изменения в коде, в том числе рефакторинг, реализовывать в порядке убывания боли от их отсутствия.

Дорога к релизу

- ~~Выбросить весь код и написать заново.~~
- Создать новый проект и вдумчиво вручную перенести в него удачные части кода.
- Неудачные — переписать с нуля.

Вопросы?

@agladysh

ag@logiceditor.com

meetup.com/Lua-in-Moscow