

# Rapid backend prototyping for a geolocation-based mobile game

With OpenResty, Redis and Docker



Alexander Gladyshev <ag@logiceditor.com>  
@agladyshev

FOSDEM 2017

# Talk plan

1. The Case
2. The Stack
3. Docker
4. HTTP API
5. Устройство игрового мира
6. Demo
7. Клиент
8. Итоги
9. Вопросы?

# About me

- Developing software since 2002
- Most of the time in gamedev
- Beyond that: high-load internet solutions, enterprise software etc. etc.
- Organizer of [meetup.com/Lua-in-Moscow](https://meetup.com/Lua-in-Moscow).  
Come to our Lua-related conference in Moscow on March 5!

# Mobile games with geolocation



# The Goals

- To try out a number of approaches to the gameplay, to generate new ideas, to figure out what is fun and what is not.
- To get as cheaply as possible the framework that would allow to iterate over gameplay variants as fast as possible.
- To figure out technical limitations of the genre in practice.

# Results

A geolocation game server-side prototype along with a rudimentary client-side was developed in less than 100 man-hours (2 calendar months)

We're rapidly iterating over the gameplay options and develop the technology part of the project.

# What is this talk about?

- This talk is about the technology part of the project,
- not about game-design
- or monetization.

It now is easier when ever to develop geolocation-based games.

I will show where to begin.

# First prototype gameplay

- The player is searching for the mobs placed on a map by walking around;
- Player has a set chance to catch the found mob.
- Caught mobs increase a stats counter in player characteristics.
- Caught mobs disappear from the map, but respawn at the same place after a set time.
- Admin users may add new mobs on the map.



# The Stack

- Server:
  - Redis,
  - OpenResty,
  - Docker.
- Client:
  - A single-page web application (in browser),
  - HTML5.

# Not Invented Here Syndrome?

Yes and No.

# Redis

- A reliable, proven solution.
- Supports geoposition out of the box:
  - GEOADD key longitude latitude member
  - GEORADIUS key longitude latitude radius m
- Useful set of primitives to store game objects in.
- "Stored procedures" in Lua.

# OpenResty

- An nginx distributive supporting Lua, Redis and many other things out of the box.
- Very fast, rather friendly and well-maintained.
- Useful both for quick prototyping and production environment.

# Docker

- Reproducible cross-platform development environment.
- Efficiently relieves the development environment pain set up.
- Can be quickly turned into a prototype of production environment (it is arguable if it is suitable for real production).
- Has to be updated to a sufficiently recent version.

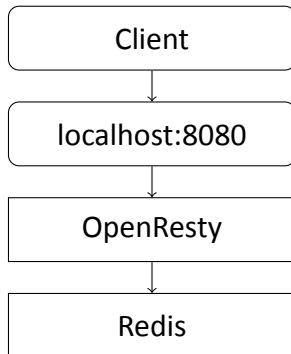
# Browser, HTML5

- Server-side during early stages of development is more important.
- Don't implement what can be imagined while play-testing, like augmented-reality combat and other frills.
- HTML5 is good to implement "quick and dirty" client.
- NB: Geolocation data access is limited, but sufficient.

# Docker: how to install on Ubuntu

- Traditionally, Ubuntu version is outdated.
- Don't do `wget | sh`.
- Install `docker` и `docker-machine` from docker's apt-repository (see manual).
- `docker-compose` is to be installed by `pip install`.
- Read the manual for the installation on other platforms.

# Docker on the developer's machine





## docker-compose.yml for development: Redis

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

## OpenResty: interesting parts of nginx.conf (reduced)

```
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/local/openresty/nginx/lualib

**COPY** nginx/lua /usr/local/openresty/nginx/lua

**COPY** nginx/static /usr/local/openresty/nginx/static

**ENTRYPOINT** /usr/local/bin/openresty-entrypoint.sh

## OpenResty: entrypoin.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 for development: 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

# API calls

- 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`.

# Demo

- Current version of the application can be found here [geo.logiceditor.com](http://geo.logiceditor.com).
- Sources are linked to at the same page.
- The very first client is always `curl`. You can try out the API using it by appending `/api/v1` to the URL of the application.

# 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 соображениям 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 геолокацию.
- 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.

Questions?

@agladysh

ag@logiceditor.com

[meetup.com/Lua-in-Moscow](https://meetup.com/Lua-in-Moscow)

Come to our Lua conference in March 5 in Moscow!