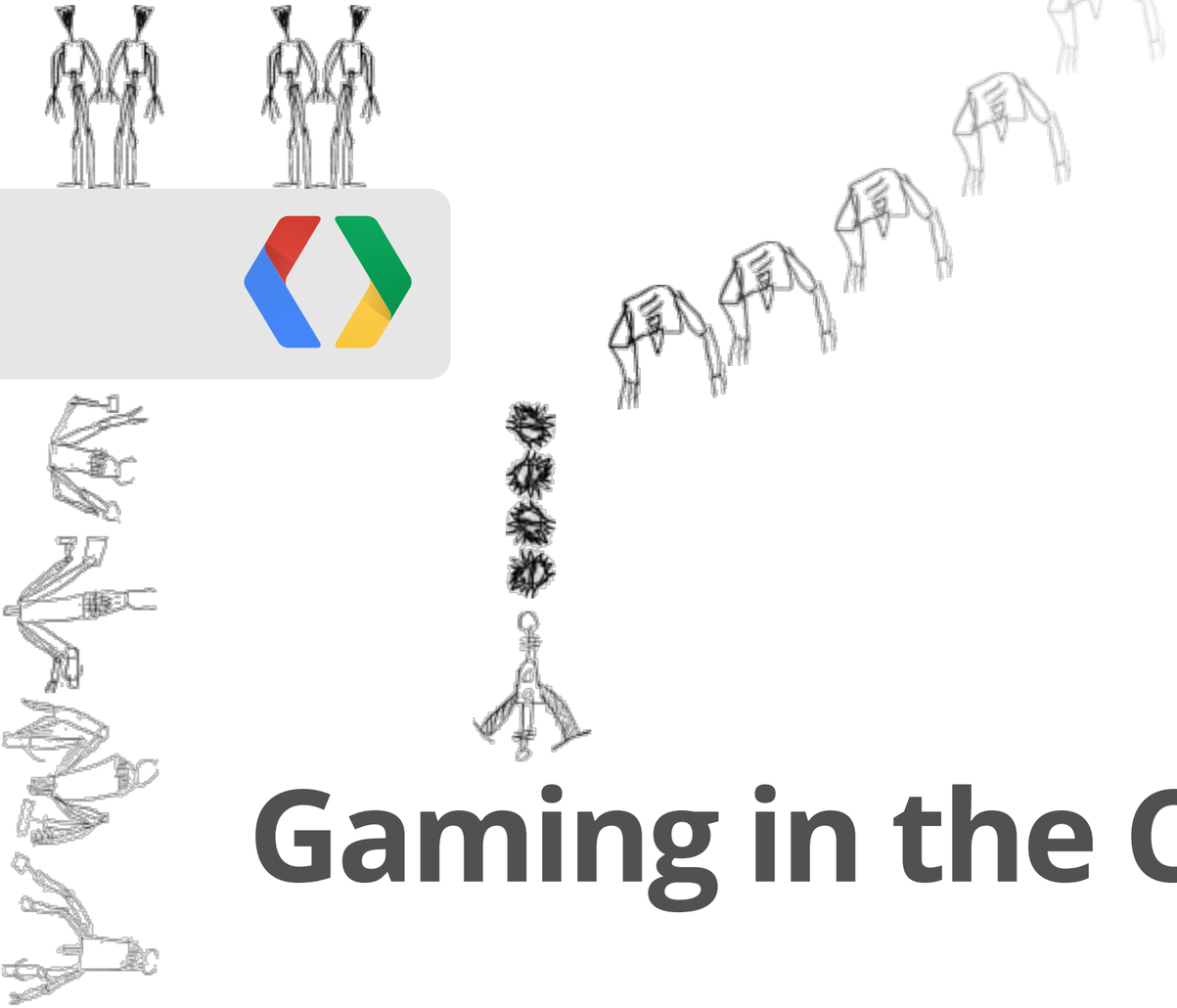
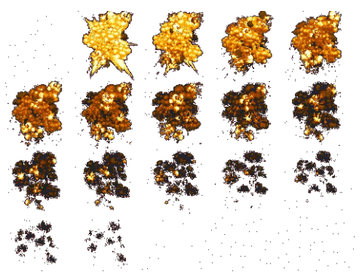
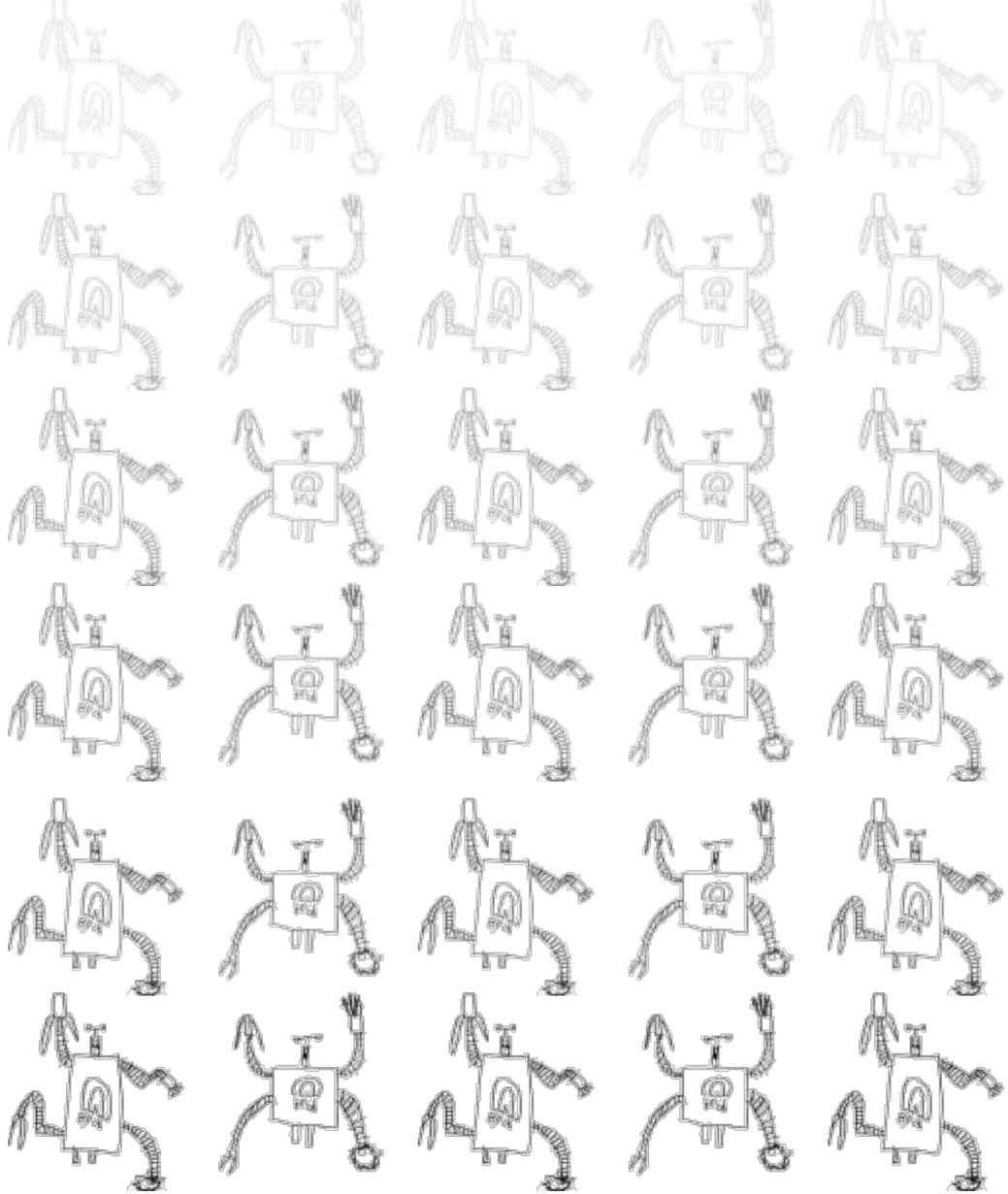


Gaming in the Cloud

Fred Sauer
Developer Advocate





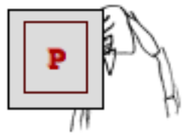
Google™ 12

hornetblast.appspot.com (2007)

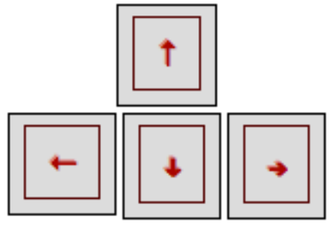
Hornet Blast

GAME OVER

Game by Fred Sauer
Pencil Artwork by Archer Sauer (age 5)
Explosion Annimation by Boris, author of JGame
Sound effects courtesy The Freesound Project
Developed with gwt-voices and Google Web Toolkit (GWT)



Pause



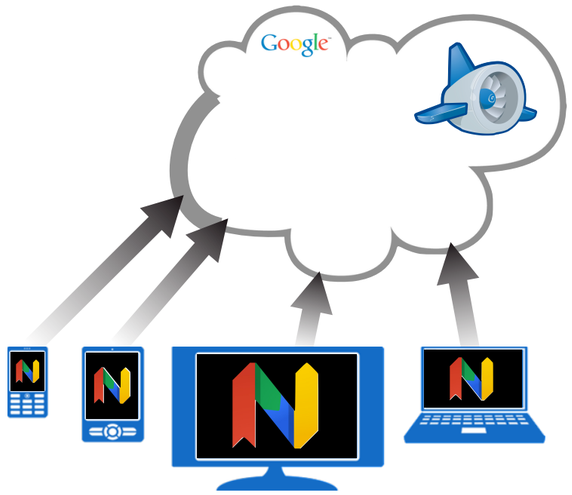
Ship movement

SPACE BAR

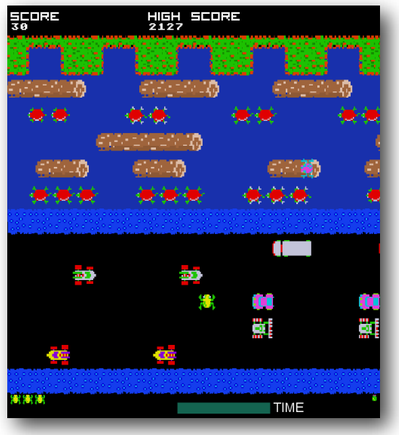
Fire / Start Game

hornetblast.appspot.com

PlayN - 1 game, N platforms

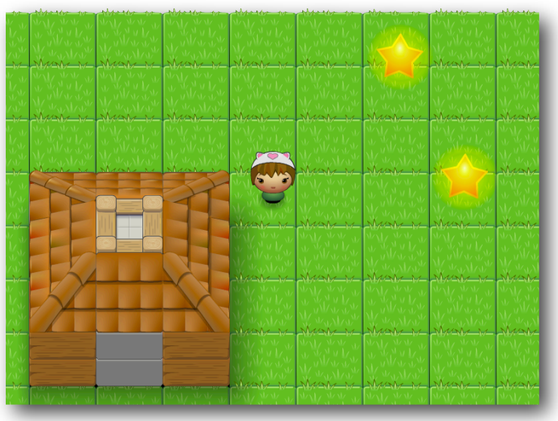


code.google.com/p/playn

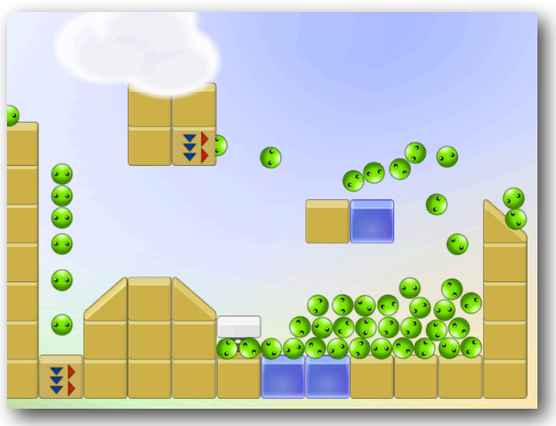


Frogger Classic

Angry Birds Chrome



Cute sample



Peas sample

Pyramid Solitaire Saga

You must be the "*master of everything*"

"Stuff" *you* need to know

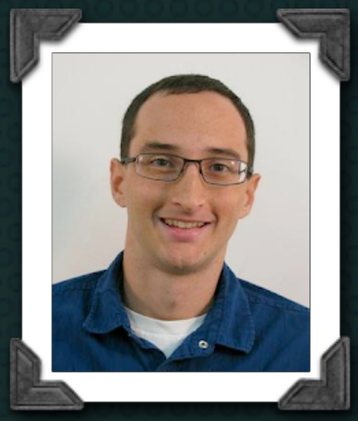
Console Developer

Web Developer

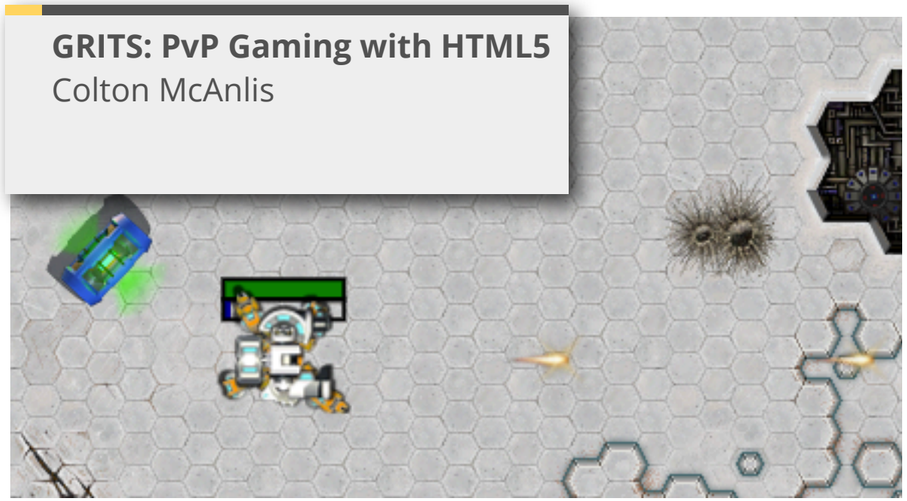
NaCl Developer

Mobile Developer

gritsgame.appspot.com (2012)



"*Grits*" development stack

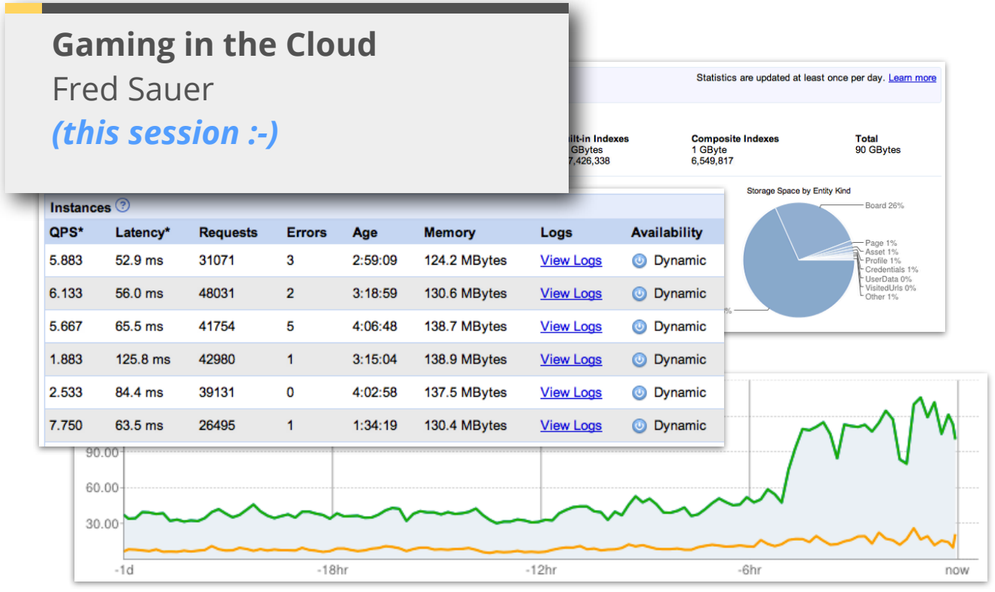


The "frontend"

- Runs in a browser
- Large potential audience

JavaScript

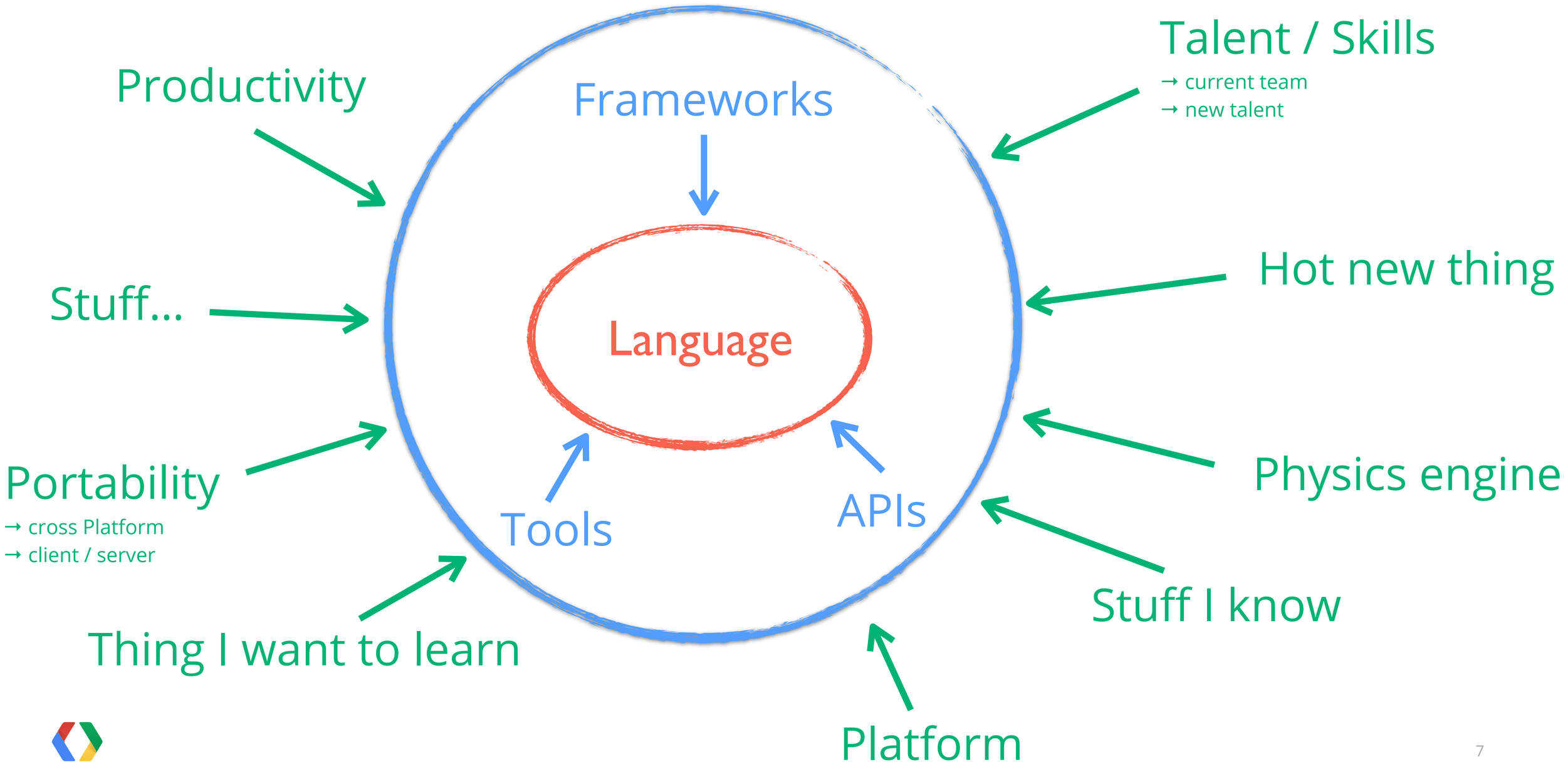
WebSockets



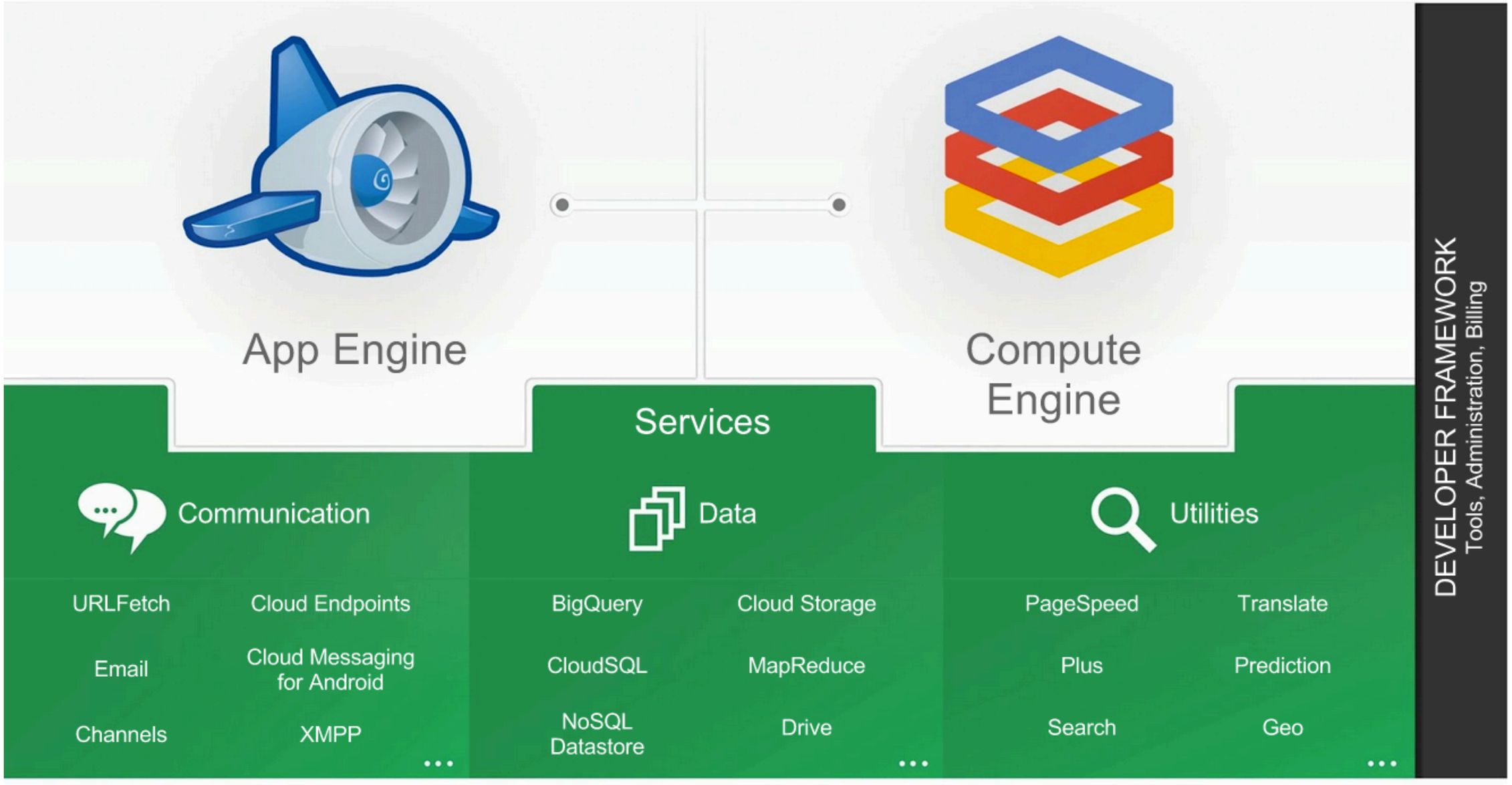
The "backend"

- Write as little as possible
- Built-in / hands-off scalability

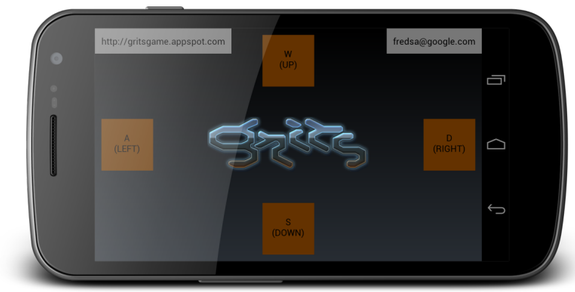
Choosing a development stack



Google Cloud Platform



"*Grits*" architecture



HTML, CSS
JavaScript

JSON

static content

frontend

JSON

'matcher' backend

App Engine

JSON

JSON

Websocket

game endpoint

game endpoint

• • •

Compute Engine

Wire protocol

Choose your wire protocol

Use someone else's plumbing

- Google App Engine Cloud Endpoints
- Google Plugin for Eclipse
  - App Engine Connected Android
  - `SharedPreferences` sync

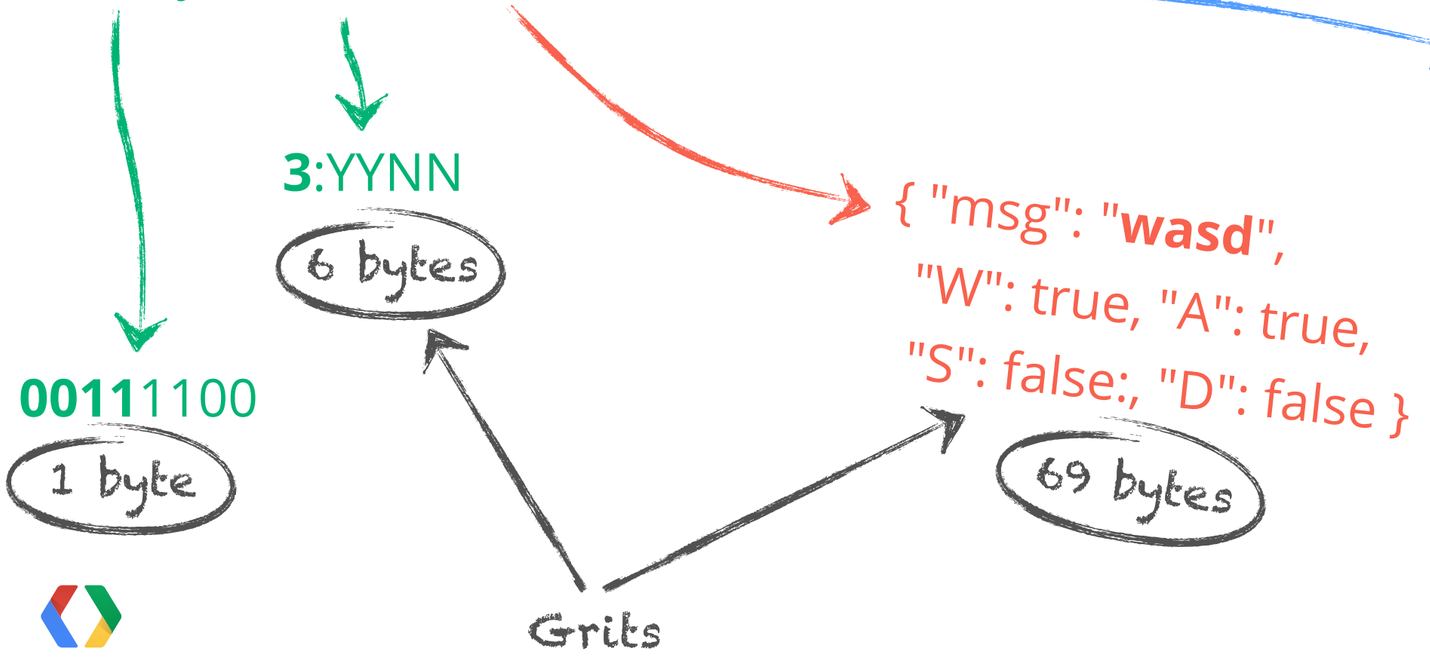
Building Mobile App Engine Backends for Android, iOS and the Web
Dan Holevoet, Christina Ilvento

Building REST APIs for Mobile with App Engine [codelab]
Dan Holevoet

Building Android Applications that Use Web APIs
Yaniv Inbar, Sriram Saroop

Create your own plumbing

- roll your own, JSON, XML



JSON for ~~robots~~ humans

```
import json

_DEBUG = True
_JSON_ENCODER = json.JSONEncoder()

if _DEBUG:
  _JSON_ENCODER.indent = 4
  _JSON_ENCODER.sort_keys = True


def tojson(python_object):
  """Helper function to output and pretty print JSON."""
  return _JSON_ENCODER.encode(python_object)

def fromjson(msg):
  """Helper function to ingest JSON."""
  try:
    return json.loads(msg)
  except Exception, e:
    raise Exception('Unable to parse as JSON: %s' % msg)
```

```
{"serverid": "SEp93hrA64fSjr5s5ttFaF0z",
"game_state": {"players": {},
"max_players": 4, "min_players": 1},
"name": "-VxV", "success": true, "gameURL":
"http://127.0.0.1:8081/-VxV", "time": 186,
"controller_host": "127.0.0.1:12345",
"port": 8081}
```

```
{
    "controller_host": "127.0.0.1:12345",
    "gameURL": "http://127.0.0.1:8081/-VxV",
    "game_state": {
        "max_players": 4,
        "min_players": 1,
        "players": {}
    },
    "name": "-VxV",
    "port": 8081,
    "serverid": "SEp93hrA64fSjr5s5ttFaF0z",
    "success": true,
    "time": 186
}
```

JSON Handler

```
class JsonHandler(webapp2.RequestHandler):
  """Convenience class for handling JSON requests."""

  def post(self, *args):
    logging.info('%s <- %s' % (self.request.path, self.request.body))
    try:
      if not self.request.body:
        raise Exception('Empty request')
      params = fromjson(self.request.body)
      r = self.handle(params, *args)
      if not r:
        raise Exception('Unexpected empty response from subclass')
      self.response.headers['Content-Type'] = 'application/json'
      self.response.write(tojson(r))
    except:
      self.response.set_status(500)
      self.response.write(traceback.format_exc())

  def handle(self, params, *args):
    raise Exception('subclasses must implement this method')
```

JSON made easy - the payoff

Before

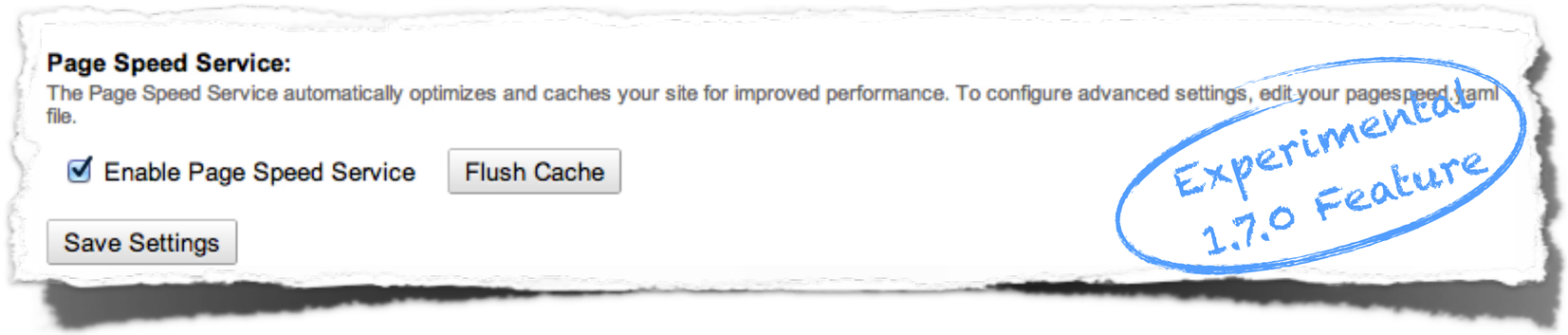
```
class GameOver(webapp2.RequestHandler):

  def post(self, params, *args):
    logging.info('%s <- %s' % (self.request.path, self.request.body))
    if not self.request.body:
      raise Exception('Empty request')
    params = fromjson(self.request.body)
    _match_maker.del_game(params['serverid'], params['name'])
    self.response.content_type = 'application/json'
    self.response.write(tojson({'success': True}))
```

After

```
class GameOver(JsonHandler):

  def handle(self, params, *args):
    _match_maker.del_game(params['serverid'], params['name'])
    return {'success': True}
```

Better, faster, more... PageSpeed Service

```
# app.yaml

handlers:
- url: …
  …

pagespeed:
  domains_to_rewrite:
    - *.gritsgame.appspot.com

  url_blacklist:
    - https://gritsgame.apppsot.com/rest/*

  enabled_rewriters:
    - CollapseWhitespace

  disabled_rewriters:
    - CombineJs
    - ProxyImages
```



HTML Rewriters

- ProxyCss - *default*
- ProxyImages - *default*
- ProxyJs - *default*
- ConvertMetaTags - *default*
- InlineCss - *default*
- InlineJs
- InlineImages
- InlinePreviewImages - *defa*
- CollapseWhitespace
- CombineHeads
- ElideAttributes
- RemoveComments
- RemoveQuotes
- LeftTrimUrls

CSS Rewriters

- CombineCss - *default*
- MoveCssToHead - *default*
- MinifyCss

Image Rewriters

- WebpOptimization - *default*
- ImageConvertToJpeg - *default*
- ImageRecompressJpeg - *defaul*
- ImageProgressiveJpeg - *default*
- ImageRecompressPng - *default*
- ImageStripMetadata - *default*
- ImageStripColorProfile - *default*
- ImageResize - *default*
- LazyloadImages - *default*
- ImageAddDimensions

JavaScript Rewriters

- CombineJs - *default*
- JsOptimize - *default*
- DeferJs

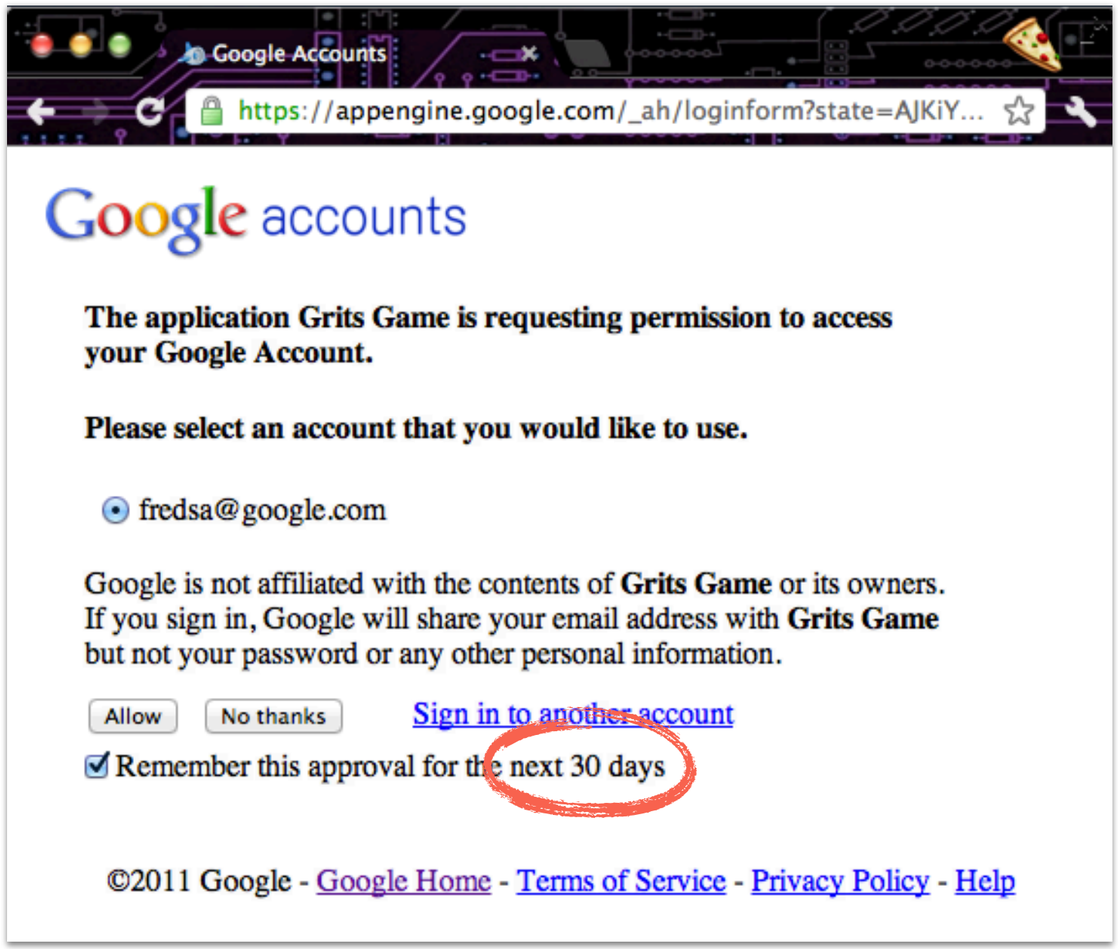
Authn & Authz

Identity & Access controls

Built-in auth - Users API

```
# app.yaml

handlers:
- url: /.*
  script: main.app
  secure: always
  login: required
```

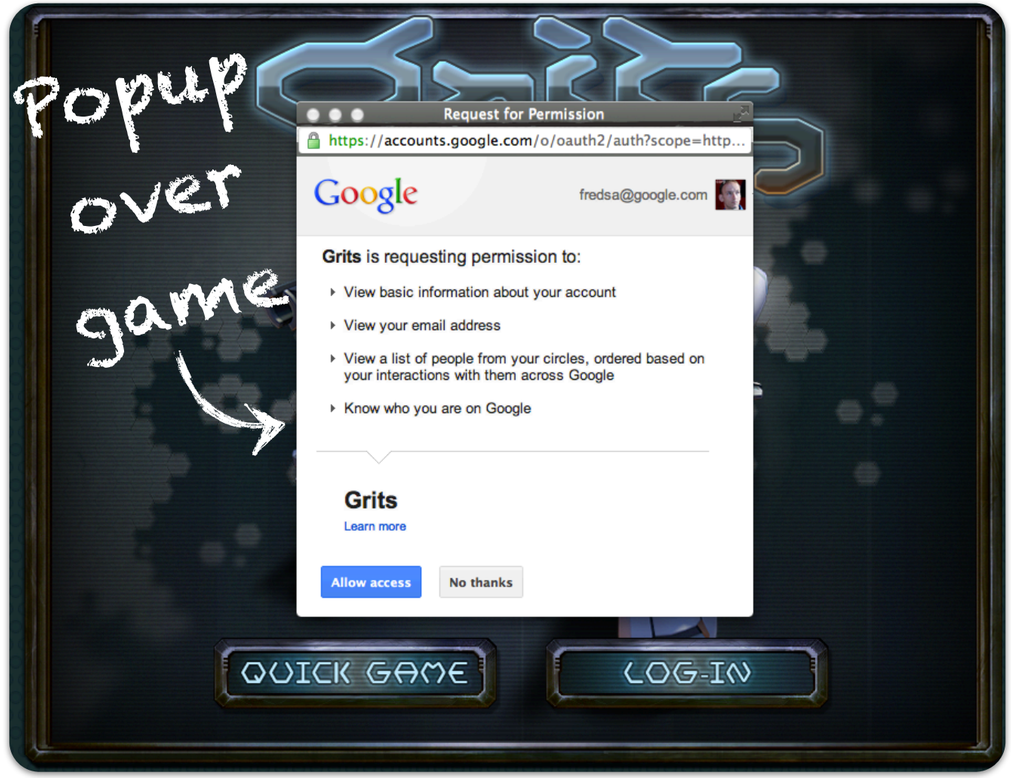
Full page redirect



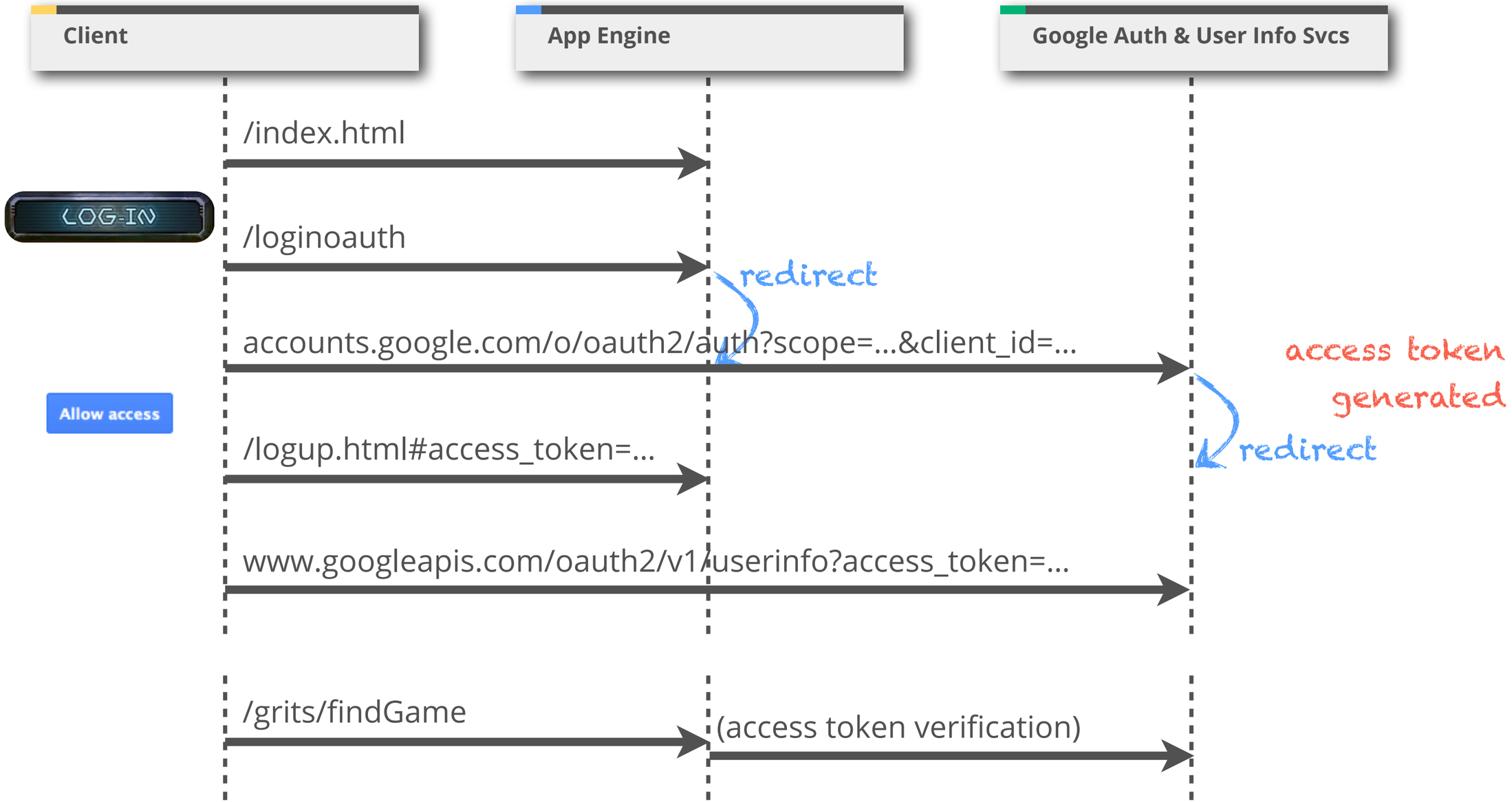

```
# main.py
from google.appengine.api import users

class MyHandler(webapp2.RequestHandler):
  def get(self):
    user = users.get_current_user()
    if user:
      body = ('Hello, %s' % user.nickname())
    else:
      body = ('Please <a href="%s">sign in</a>.' % users.create_login_url('/'))

    self.response.write('<html><body>%s</body></html>' % body)
```

Built-in auth - OAuth2

```
# Client obtains and passes in the OAuth2 access token
Authorization: Bearer ya29.XAHES6ZT4w77FecXjZu4ZWrkTSX…

# app.yaml
handlers:
- url: /rest/.*
  script: main.app
  secure: always

# main.py
from google.appengine.api import oauth

SCOPE = 'https://www.googleapis.com/auth/userinfo.email'
user = oauth.get_current_user(SCOPE)

# returns the same value as the Users API: users.get_current_user().user_id()
id    = user.user_id() # 813856575646671226907
email = user.email()   # fredsa@google.com

# Debugging your access tokens
$ curl https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ya29.XAHES6ZT4w77FecXjZu4Z…
```

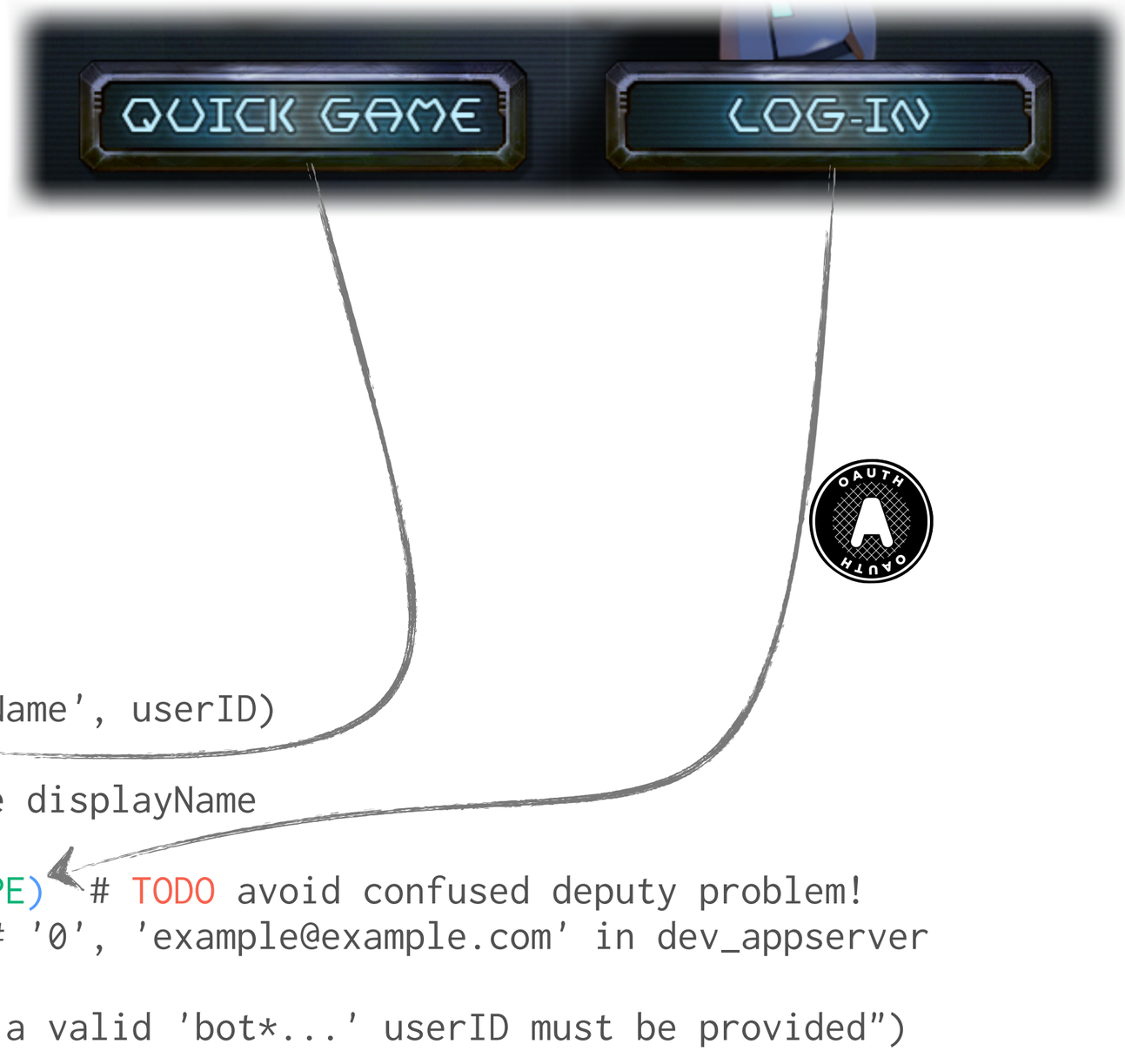


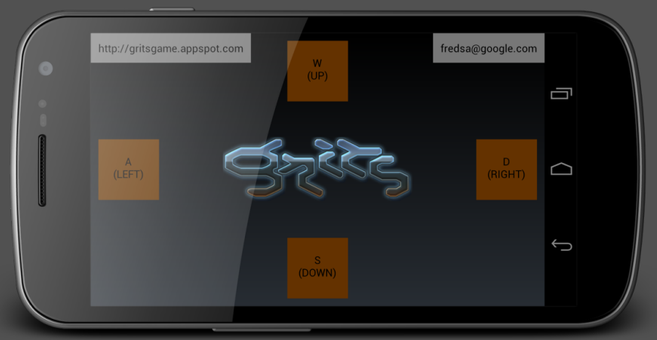
"Grits" OAuth2 flow



"*Grits*" Authentication

```
class GritsService(FrontendHandler):

  def post(self, fcn):
    userID, displayName = self.determine_user()
    …


class FrontendHandler(webapp2.RequestHandler):

  def determine_user(self):
    userID = self.request.get('userID')
    if _IS_DEVELOPMENT:
      return userID, self.request.get('displayName', userID)
    if userID and userID.startswith('bot*'):
      return userID, userID # use userID as the displayName
    try:
      user = oauth.get_current_user(_EMAIL_SCOPE)  # TODO avoid confused deputy problem!
      return user.user_id(), user.nickname()  # '0', 'example@example.com' in dev_appserver
    except oauth.OAuthRequestError:
      raise Exception("OAuth2 credentials -or- a valid 'bot*...' userID must be provided")
```



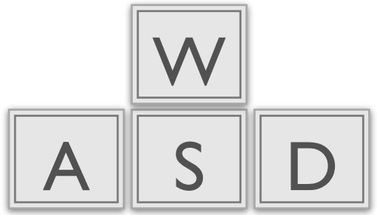


It's a really nice looking accelerometer.

It also makes phone calls.

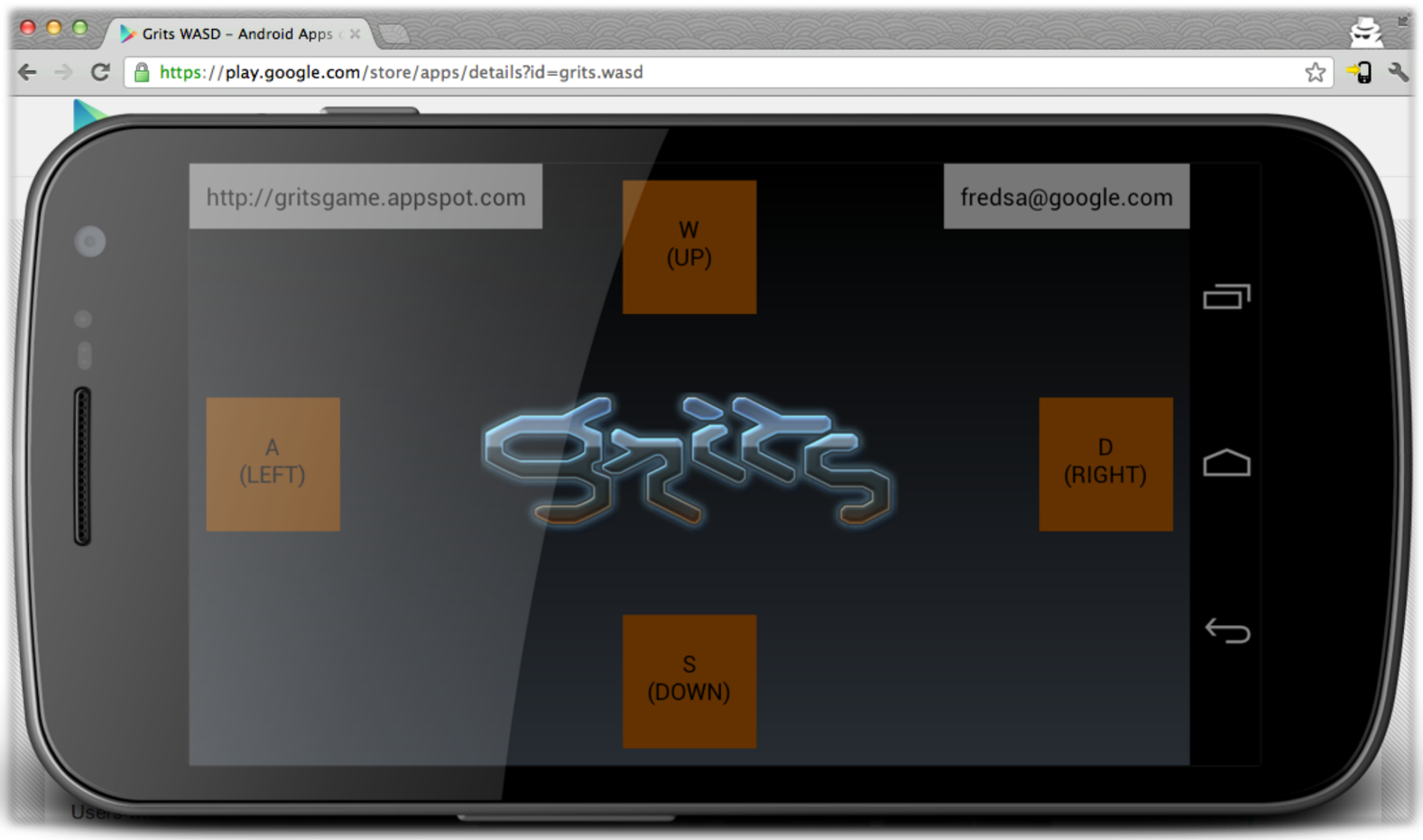
What's that in your pocket?

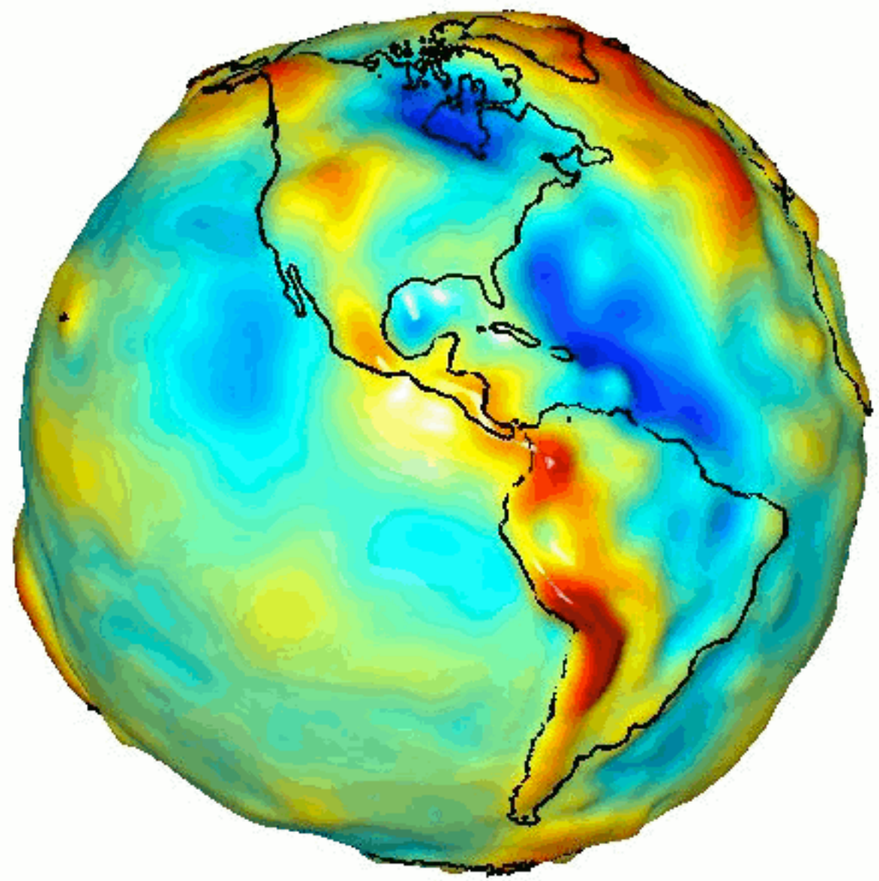
Grits WASD



DEMO TIME

~9.8 m/s^2





android-ui-utils.googlecode.com/hg/asset-studio/dist/device-frames.html

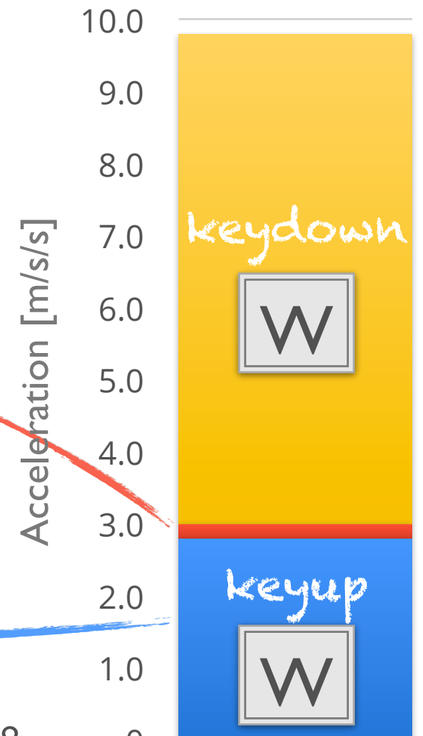
en.wikipedia.org/wiki/File:GRACE_globe_animation.gif

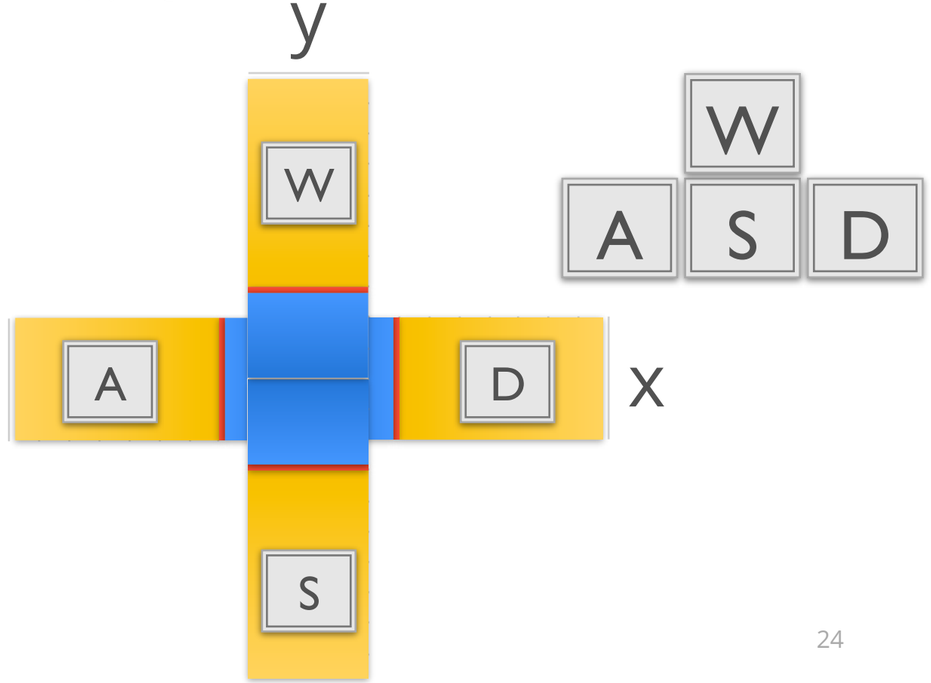
Measuring gravity - Android

```
class SensorListener implements SensorEventListener {
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {}

    @Override
    public void onSensorChanged(SensorEvent event) {
      if (event.sensor.getType() != Sensor.TYPE_ACCELEROMETER) return;
      assert display.getRotation() == Surface.ROTATION_90; // fixed landscape via manifest
      float sensorX = -event.values[1];
      float sensorY = event.values[0];
      updateDirections(sensorX, sensorY);
    }

    public void start() { // called from Activity#onResume()
      sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_GAME);
    }

    public void stop() { // called from Activity#onPause()
      sensorManager.unregisterListener(this);
    }
}
```

Directional input - Android

```
private void updateDirections(float sensorX, float sensorY) {

  if (sensorX > SENSOR_THRESHOLD) { // x > 3
    left = true;
  } else if (sensorX < SENSOR_THRESHOLD - SENSOR_STICKINESS) { // x < 2.8
    left = false;
  }

  if (sensorX < -SENSOR_THRESHOLD) { // x < -3
    right = true;
  } else if (sensorX > -(SENSOR_THRESHOLD + SENSOR_STICKINESS)) { // x > -2.8
    right = false;
  }

  …

  leftView.setBackgroundColor(left ? ORANGE : BROWN);
  rightView.setBackgroundColor(right ? ORANGE : BROWN);

  …
}
```





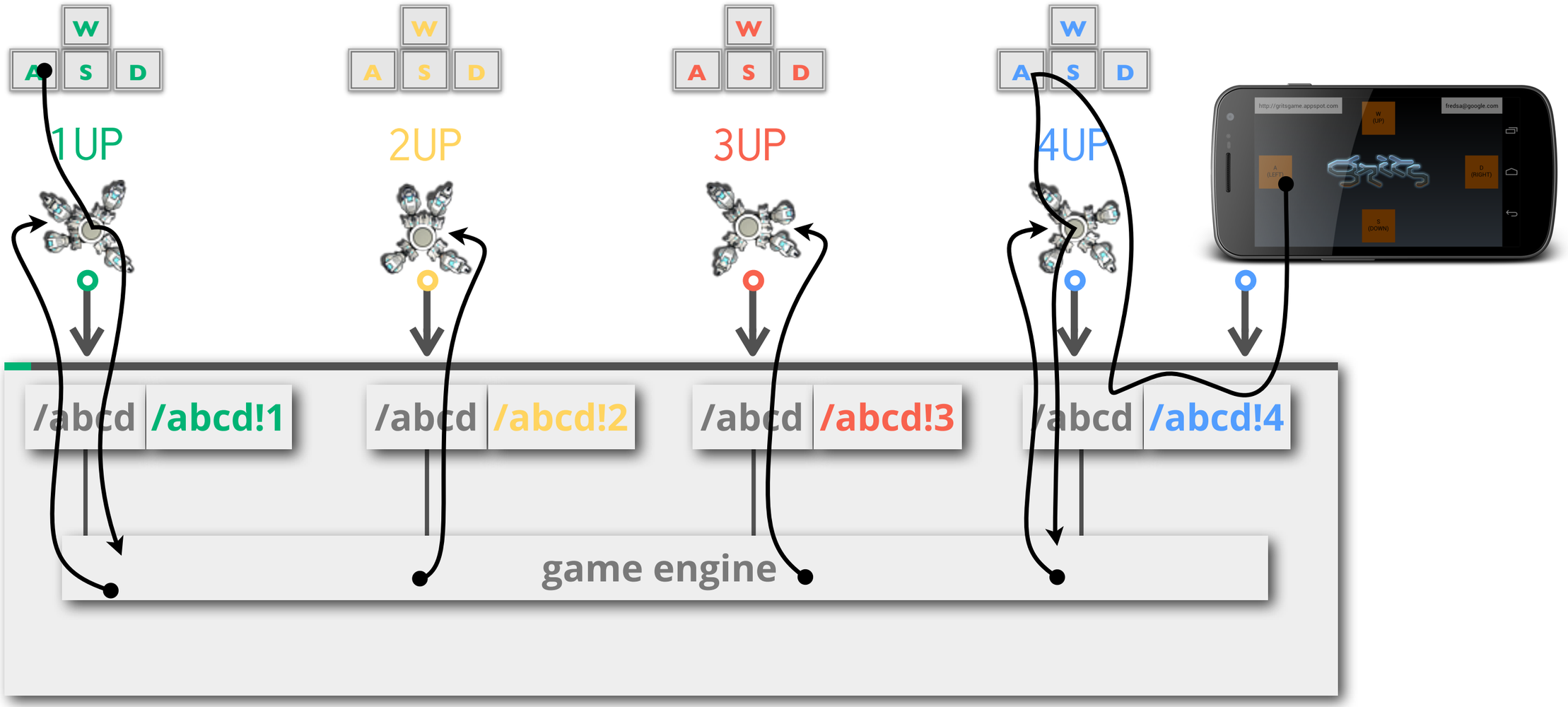
WasdActivity

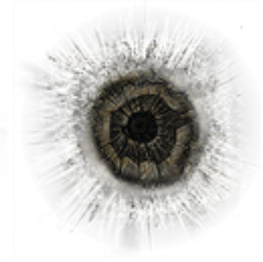
```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

  prefs = getSharedPreferences("grits", MODE_PRIVATE);
  accountManager = AccountManager.get(this);
  sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
  display = ((WindowManager) getSystemService(WINDOW_SERVICE)).getDefaultDisplay(); //rotation

  setContentView(R.layout.wasd);
  upButton = (Button) findViewById(R.id.button_up);
  leftButton = (Button) findViewById(R.id.button_left);
  rightButton = (Button) findViewById(R.id.button_right);
  downButton = (Button) findViewById(R.id.button_down);
  …

  sensorListener = new SensorListener(); // our custom listener


  …
}
```

Don't forget to pause your sensor listener

```
@Override
protected void onResume() {
  super.onResume();
  sensorListener.start();
}

@Override
protected void onPause() {
  super.onPause();
  sensorListener.stop();
}
```

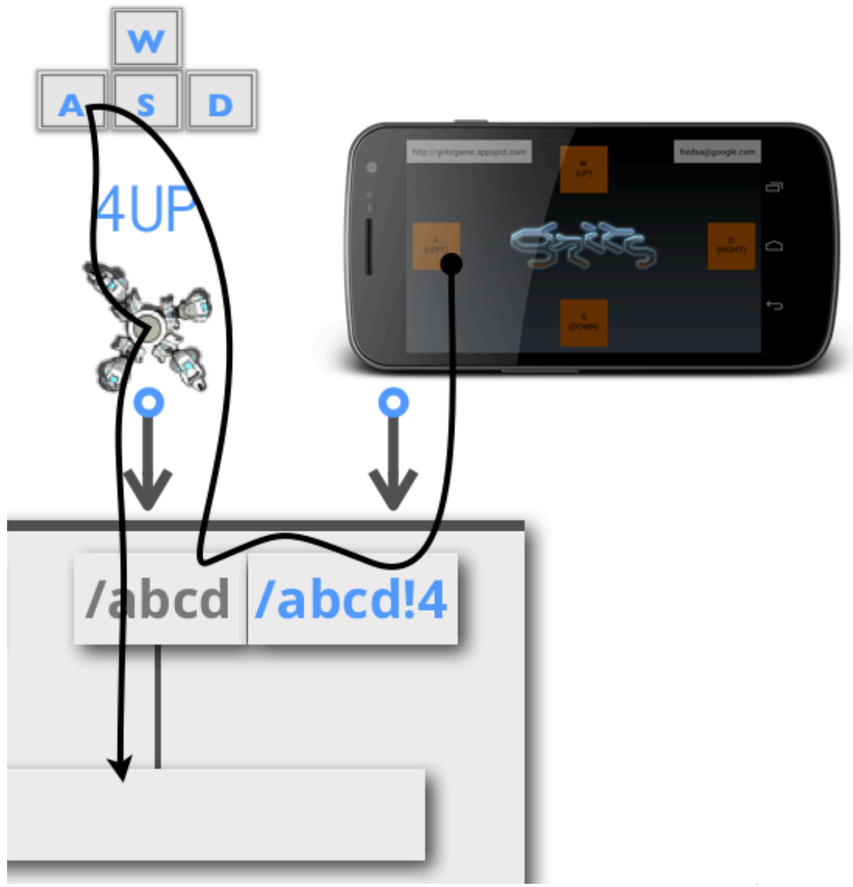
Let's hack in a game controller...



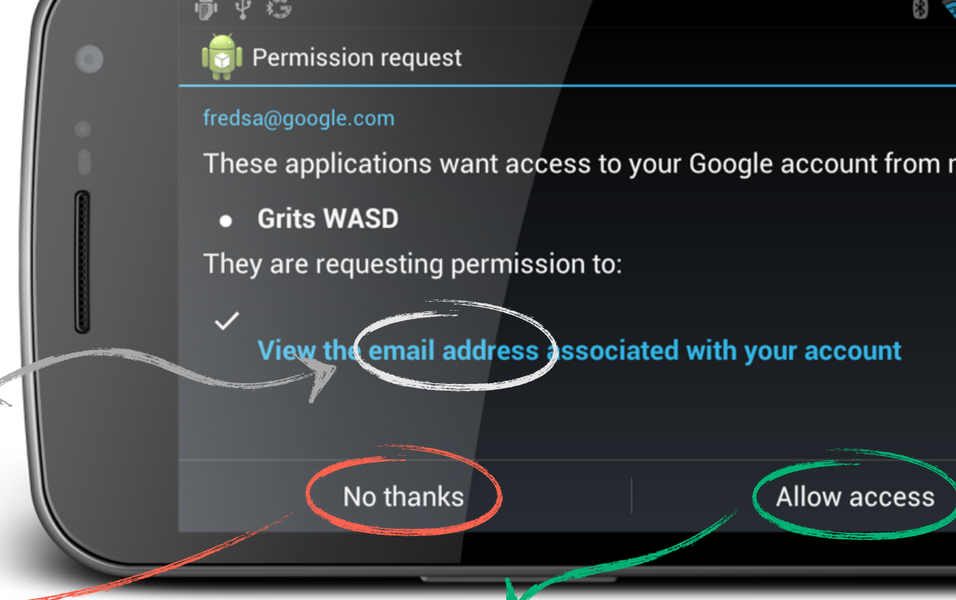


Injecting synthetic key events

```
wasd: function(msg) {
  // e.g. msg={"W": false, "A": true, "S": false, "D": false}

  for (c in msg) {
    evt=document.createEvent("Events");
    evt.initEvent(msg[c] ? 'keydown' : 'keyup', true, true);
    evt.view = window;
    evt.keyCode = c.charCodeAt(0);
    evt.charCode = c.charCodeAt(0);
    window.dispatchEvent(evt);
  }
}
```



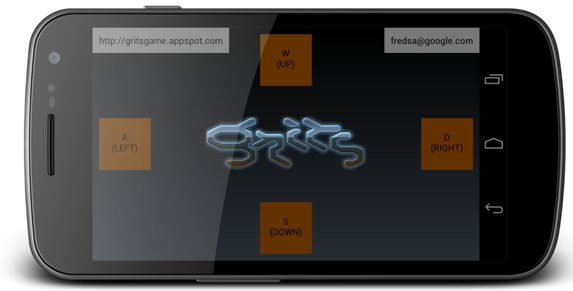
OAuth2 recipe - Android




```
AccountManager acctMgr = AccountManager.get(this);
Account acct = acctMgr.getAccountsByType("com.google")[0];

acctMgr.getAuthToken(acct,
    "oauth2:https://www.googleapis.com/auth/userinfo.email",
    null, this, new AccountManagerCallback<Bundle>() {
      public void run(AccountManagerFuture<Bundle> future) {
        try {
          String accessToken = future.getResult().getString(AccountManager.KEY_AUTHTOKEN);
          callMyAppEngineRestApiWithHttpHeader("Authorization: Bearer " + accessToken);
        } catch (OperationCanceledException e) {
          Log.w(TAG, "The user has denied you access to this scope");
        } catch (Exception e) {
          Log.w(TAG, "Unexpected exception", e);
        }
      }
    }, null);

// TODO Avoid en.wikipedia.org/wiki/Confused_deputy_problem
// TODO Invalidate expired tokens
// acctMgr.invalidateAuthToken(acct.type, accessToken);
```

"*Grits*" Services & APIs

"*Grits*" App Engine services & APIs



App Engine

PageSpeed Service
(your content)

Appservers (frontends)
(your code)

High Replication Datastore
(your data)

OAuth2 API
(your users)

...
...

Persistence / State:
High scores
World state
Item purchases

Computation / Services:
Matchmaking
Game verification
Authn & Authz
PageSpeed

TODO:
Saved games
Leaderboard
VM Management

Managing Google Compute Engine Virtual Machines Through Google App Engine
Adam Eijdenberg, Alon Levi

High Replication Datastore

Synchronous, multi-datacenter replication

More 9s Please: Under The Covers of the High Replication Datastore
Alfred Fuller, Matt Wilder

(Google I/O 2011)

```
# db_api.py

# avoid storing credentials in source code
class Config(ndb.Model):
  client_id = ndb.StringProperty(indexed=False)
  client_secret = ndb.StringProperty(indexed=False)
  api_key = ndb.StringProperty(indexed=False)

class User(ndb.Model):
  # we use str(userID) as the __key__
  displayName = ndb.StringProperty(indexed=False)
  createDate = ndb.DateTimeProperty(indexed=False)

  numWins = ndb.IntegerProperty(indexed=False)
  numLoss = ndb.IntegerProperty(indexed=False)

  credits = ndb.IntegerProperty(indexed=False)

  # e.g. ["1234", "1374", "4378", …]
  virtualItems = ndb.StringProperty(repeated=True, indexed=False)
```

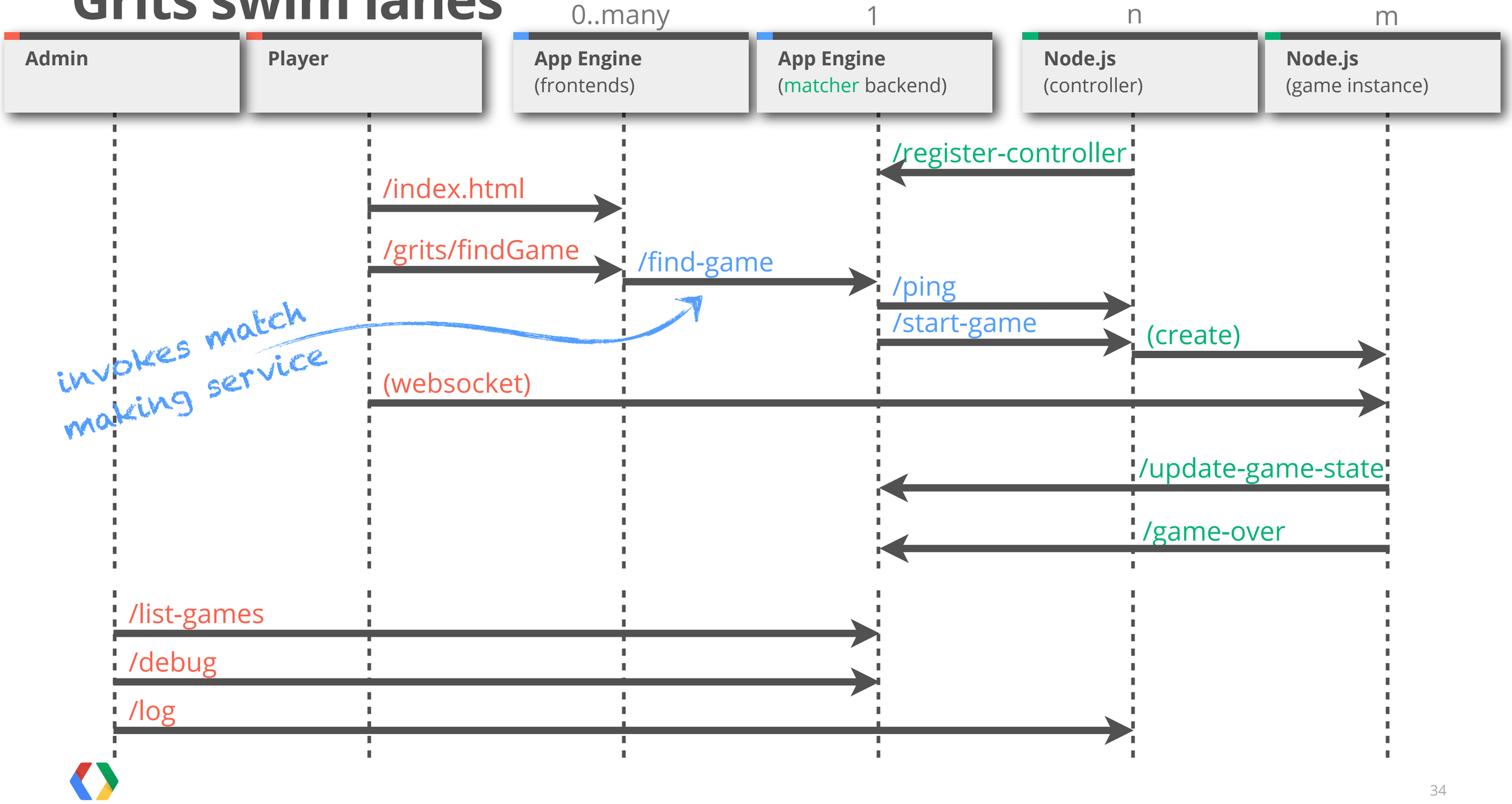
Config
client_id : ...
client_secret: ...
api_key : ...

User
Player : bot*412
Wins : 𝍸 ||
Losses : |||
Credits: 32,910

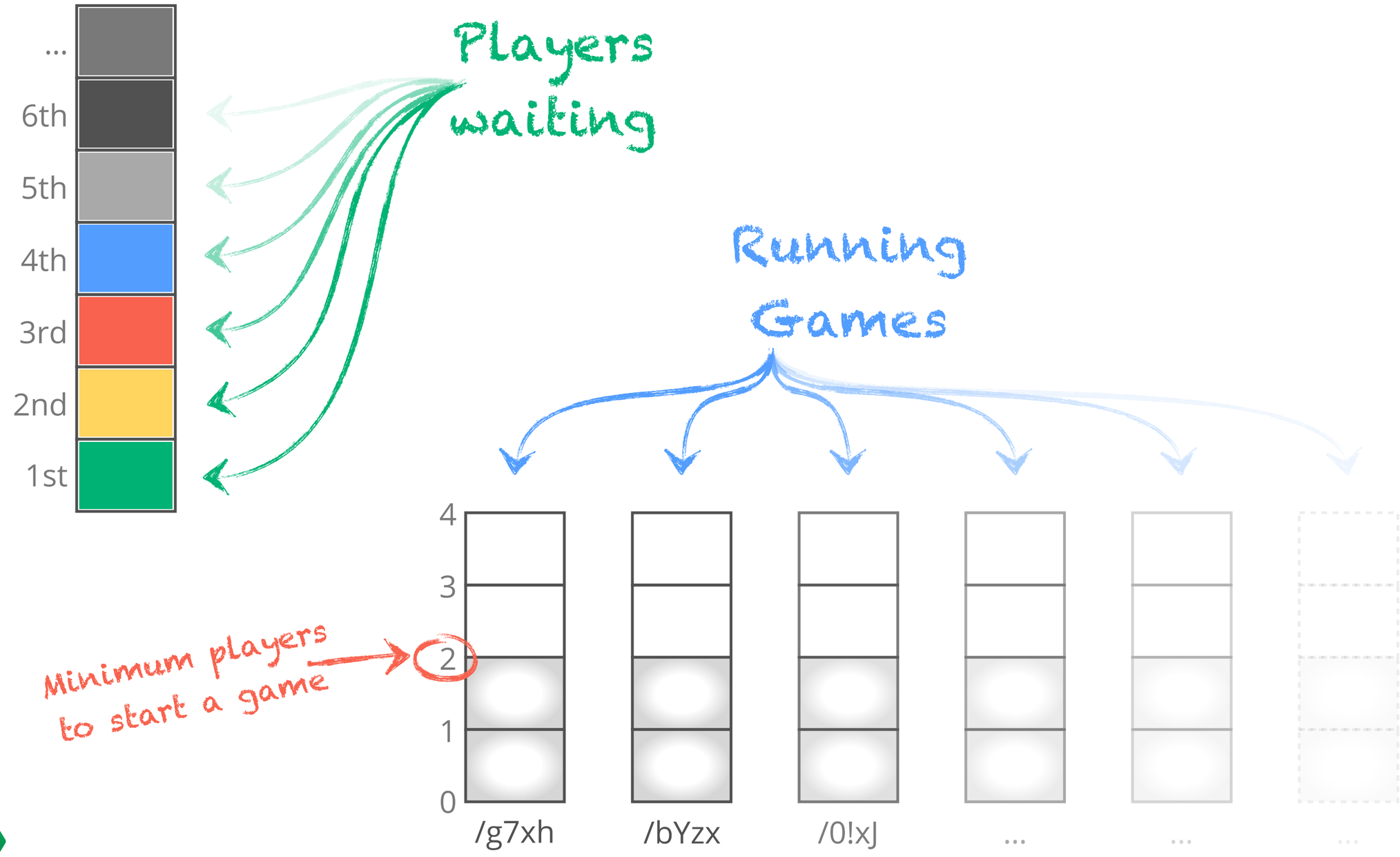
Herding cats bots

Clients coordinating with each other in the cloud

Grits swim lanes

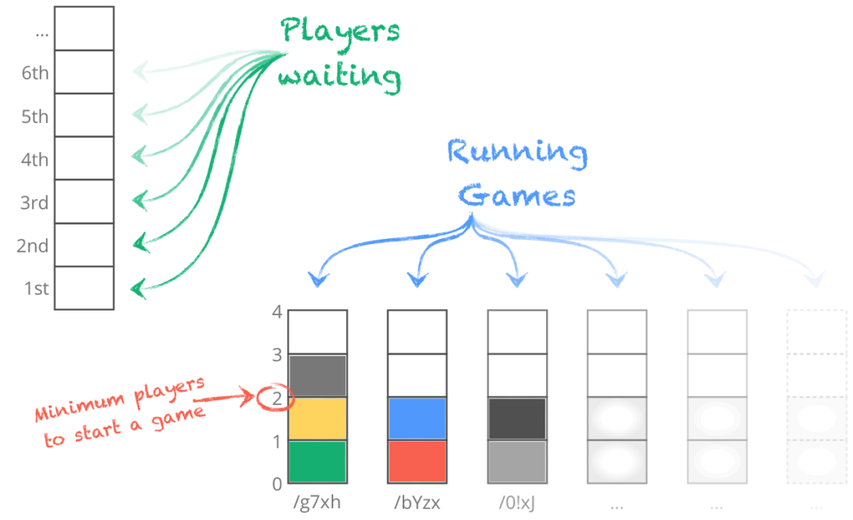


Matchmaking in cyberspace



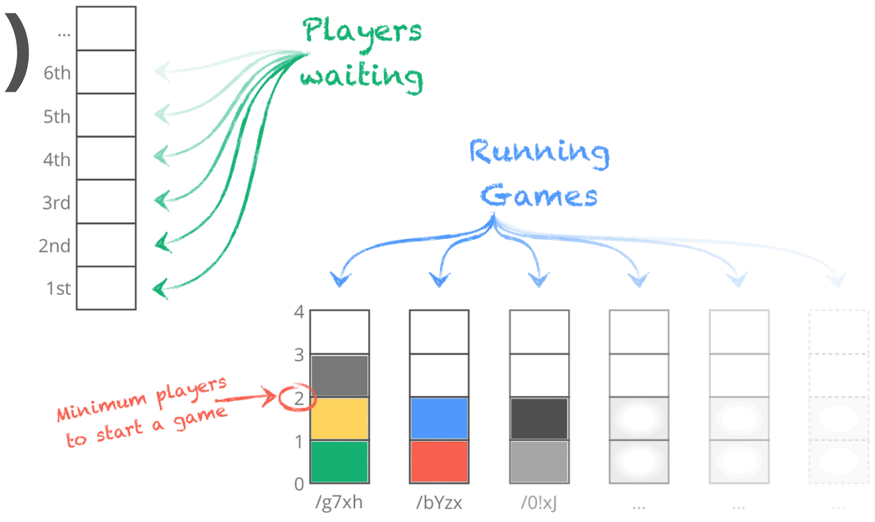
Matchmaking in cyberspace

```
def make_matches(self):
  if not self._players_waiting:
    return

  # pass 1
  players_needed_for_next_game = \
      self.make_matches_min_players()

  # pass 2
  if self._players_waiting:
    self.make_matches_max_players()

  return players_needed_for_next_game
```



Matchmaking in cyberspace (cont'd)

```
def make_matches_min_players(self):
  players_needed_for_next_game = -1  # sentinel value
  for server_struct in self._game_servers.itervalues():
    for game in server_struct['games'].itervalues():
      game_state = game['game_state']
      players_in_game = game_state['players']
      player_goal = int(game_state['min_players'])
      players_needed = player_goal - len(players_in_game)
      if not players_needed:
        continue
      if len(self._players_waiting) >= players_needed:
        # let's get this party started
        while len(players_in_game) < player_goal:
          self._add_player(self._players_waiting.pop(0), game)
      elif (players_needed_for_next_game == -1
            or players_needed < players_needed_for_next_game):
        players_needed_for_next_game = players_needed
return players_needed_for_next_game
```

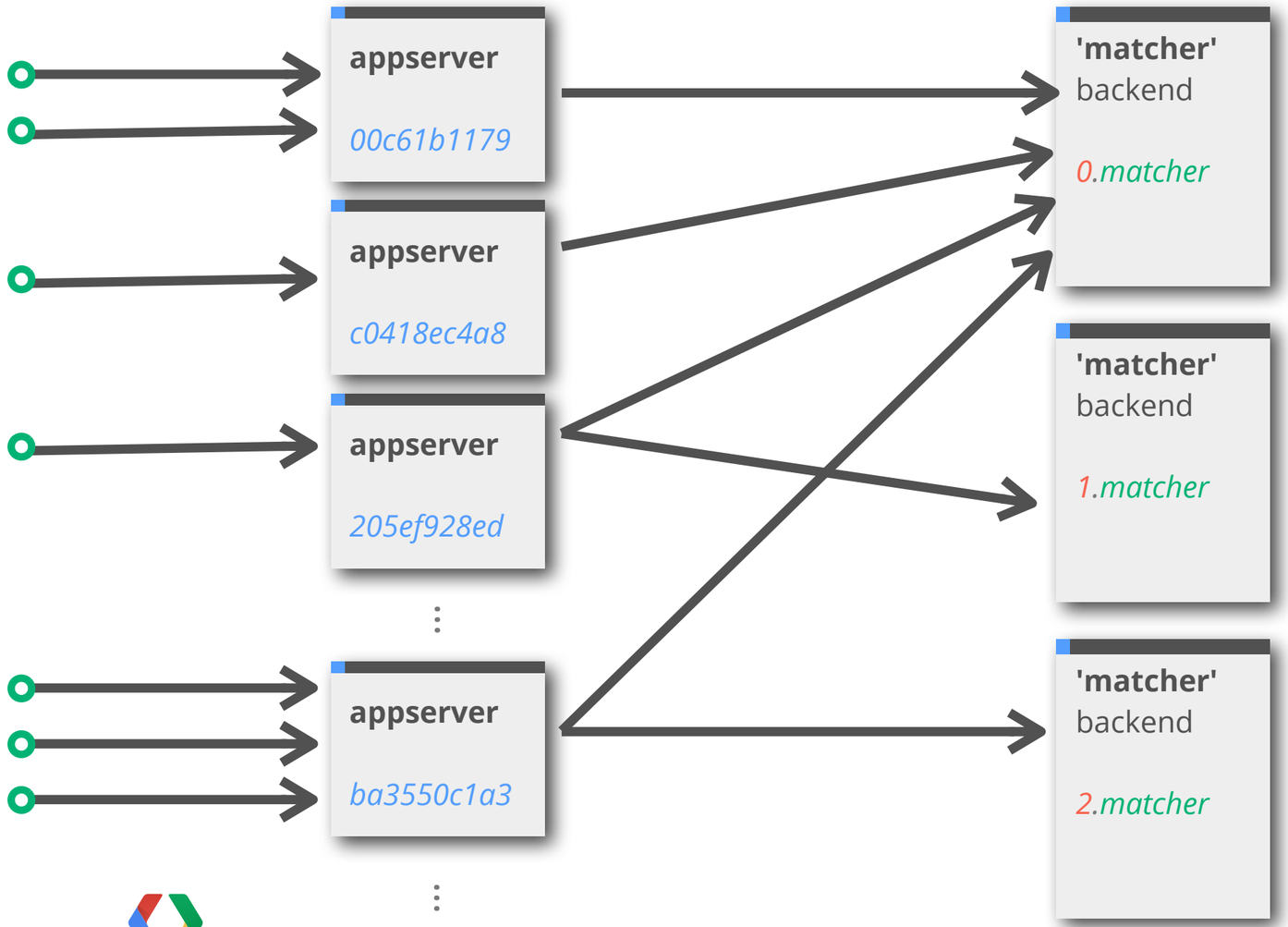


Scale it up

Dynamic "sticky" backends

"frontends"
gritsgame.appspot.com

"backends"
matcher.gritsgame.appspot.com
{0,1,2}.matcher.gritsgame.appspot.com




```
# backends.yaml
backends:
- name: matcher
  options: public, dynamic
  instances: 3
```

```
# main.py
from google.appengine.api import backends

n = backends.get_instance()
new_url = backends.get_url(instance=n)
r = {'new_url': new_url}
response.write(tojson(r))
```

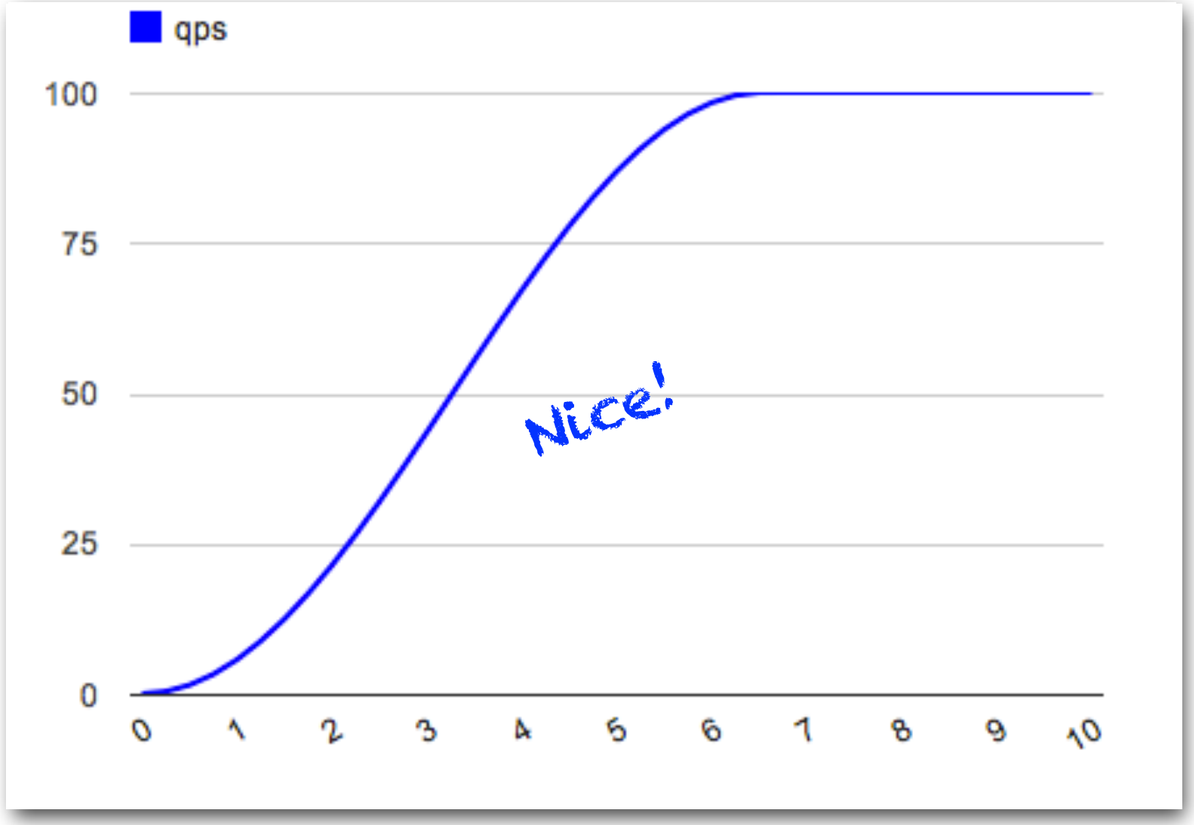
Perform load tests

Traffic should be representative of real user traffic

Ramp up over a period of several minutes

Basic Recipe:

- Send steady 5 qps traffic to your site
  - Uncover application bottlenecks
- Monitor for 15-20 minutes and expect...
  - Zero quota denials
  - Low error rate
  - Tasks queues are keeping up
  - No elevated datastore contention
- Slowly ramp up traffic to 2x qps
  - Uncover next application bottleneck, if any
- Repeat until serving desired qps
- Relax: plan your next feature



qps = requests / second

Don't DoS yourself

~~Clients check in at a specific time of day~~

Spread out load:
- Randomize client checkin

~~Clients retry failures after a fixed interval~~

Avoid a self inflicted Denials-of-Service (DoS) on your app after a failure:
- Use exponential back-off: 1s, 2s, 4s, 8s, 16s, ...
- Use a fuzz factor: randomize retry times

Scalability basics

Static vs. dynamic content

- Static handlers (`app.yaml` / `appengine-web.xml`)
- Google Cloud Storage, Blobstore API

Use `Cache-Control`

- Free lunch

Stacked caching

- Runtime → Memcache / Backend → Datastore

Entity group writes

- At least 1 write/sec

Monotonically increasing keys

- Monotonically increases indexes
- Random, not db.allocate_ids() / DatastoreService#allocateIds()

Scalability basics (cont'd)

RTT, # HTTP connections

PageSpeed

Multi-threaded

- `threadsafe: true` / `<threadsafe>true</threadsafe>`
- Python 2.7 / Java only

Async RPCs

- Async URL Fetch
- Async Datastore

Task Queue

- Do it later

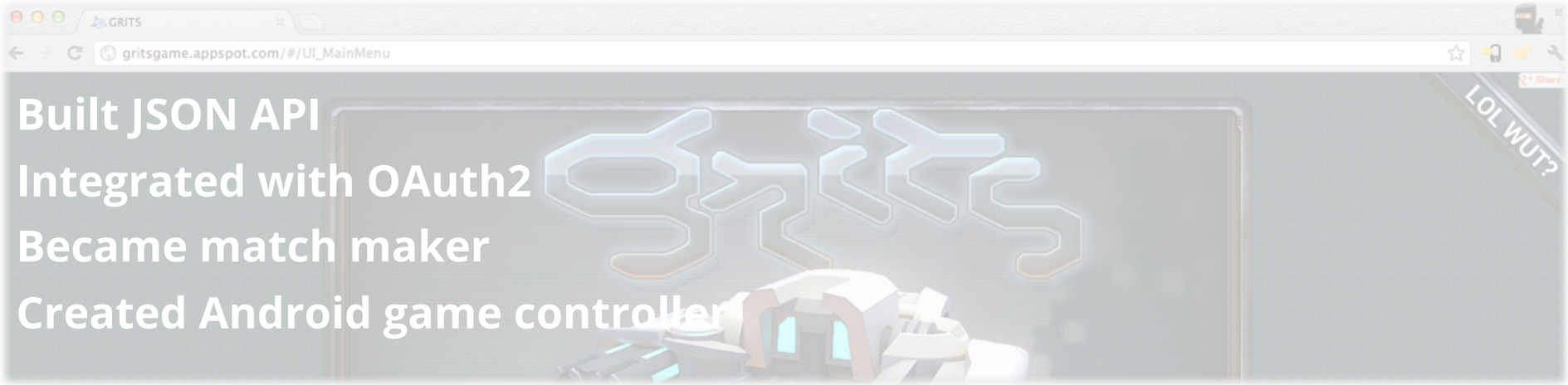
Eventual consistency

- Non-ancestor queries

M/S → HRD

Summary

What you saw...

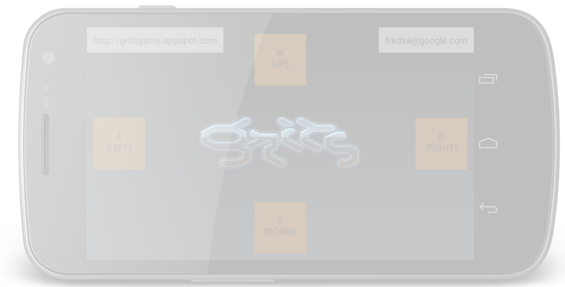


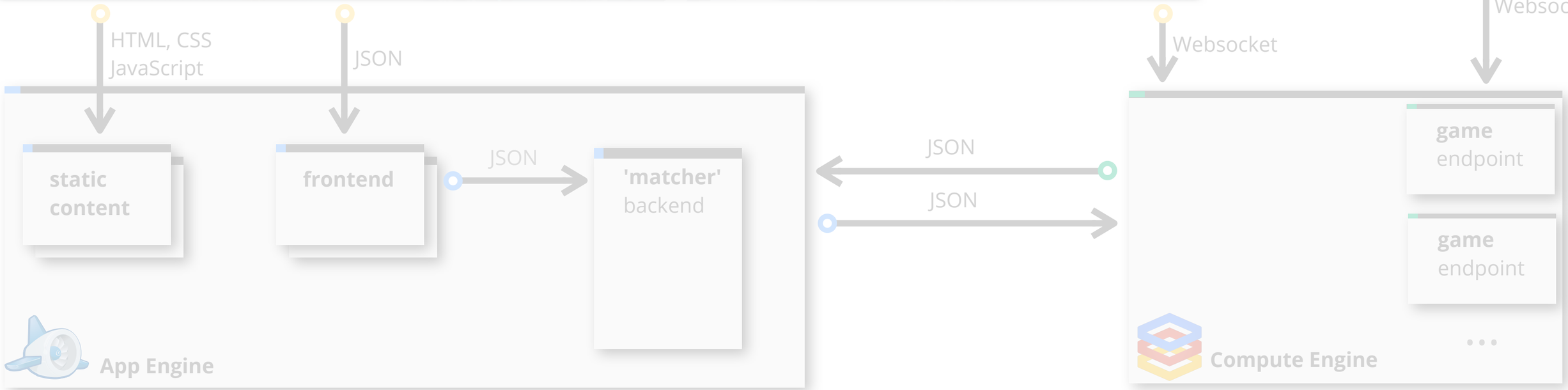
Built JSON API

Integrated with OAuth2

Became match maker

Created Android game controller





Come chat during Office Hours

Thank You!

Fred Sauer
fredsa@google.com

gritsgame.appspot.com

developers.google.com/appengine
developers.google.com/compute

code.google.com/p/gritsgame
code.google.com/p/playn