Google Earth

April 9 2015
Google Earth

• Download and install Google Earth
  – If you haven’t done so, a TA will help you now

• Download CIT.kml from the course webpage
  – Open the file on Google Earth
  – Open the file using
    • TextEdit (Mac)
    • Notepad (Win)

In groups, do Task 1
Hmm, let’s talk about the color in the KML file
KML Color Codes

O = Opacity
B = Blue
G = Green
R = Red

Hexadecimal (16 digits): 0-9,a-f
00 01 ... 08 09 0a 0b 0c 0d 0e 0f 10 11 ... fe ff
KML Color Codes

Go to http://www.color-hex.com

Color Hex Color Codes

Color-hex gives information about colors including color models (RGB, HSL, HSV and CMYK), Triadic colors, monochromatic colors and colors calculated in color page. Color-hex.com also generates a simple css code for the selected color. Html element samples are also shown below the color detail page. Simply type the 6 digit color code in the box above and hit enter.

Users Latest Favorite Colors

#f7d1d7  #ffffff  #aaffda  #9fdd1  #95c1d0  #b9f6df
KML Color Codes

Go to [http://www.color-hex.com](http://www.color-hex.com)

Color Hex Color Codes

Color-hex gives information about colors including color models (RGB, HSL, HSV and CMYK), Triadic colors, monochromatic colors and colors calculated in color page. Color-hex.com also generates a simple css code for the selected color. Html element samples are also shown below the color detail page. Simply type the 6 digit color code in the box above and hit enter.

Users Latest Favorite Colors

- #f7d1d7
- #ffffff
- #aaffda
- #9fdd1d
- #95c1d0
- #b9f6df
KML Color Codes

Go to http://www.color-hex.com

Color Hex Color Codes

Color-hex gives information about colors including color models (RGB,HSL,HSV and CMYK), Triadic colors, monochromatic colors and colors calculated in color page. Color-hex.com also generates a simple css code for the selected color. Html element samples are also shown below the color detail page. Simply type the 6 digit color code in the box above and hit enter.

Users Latest Favorite Colors

#f7d1d7  #ffffff  #aaffda  #9fdd1  #95c1d0  #b9f6df
KML Color Codes

#00b00e Color Hex

Use this in your KML

#00B00E
(0,190,239)
Google Earth

- Now open CIT.kml in a text editor and try changing the following:
  - Name
  - Description
  - Size of Pin
  - Coordinates
  - Color ffefbe00

In groups, do Task 2

http://geographyworldonline.com/tutorial/instructions.html
Last class...

• Go to:
  – [https://maps.googleapis.com/maps/api/geocode/xml?address=115+Waterman+Street,+Providence,+RI](https://maps.googleapis.com/maps/api/geocode/xml?address=115+Waterman+Street,+Providence,+RI)

• This is Google letting you:
  – Give an address
  – Get back an XML containing latitude/longitude

```
<location>
  <lat>41.8271609</lat>
  <lng>-71.3995390</lng>
</location>
```
Last class...

• Go to:

• This is Google letting you:
  – Give latitude/longitude
  – Get back an XML with address information
Application Programming Interfaces

• Web services *expose functionality* via *XML*
• Or via something called *JSON*
  – Even *more convenient*!

• The Google Earth example is an example of an API
  – Application Programming Interface
Let’s try the same with JSON

• Go to:

• This is Google letting you:
  – Give latitude/longitude
  – Get back a JSON response with address information

  – Let’s check its contents
{ "results" : [ 
{
   "address_components" : [ 
{ 
   "long_name" : "Mountain View",
   "short_name" : "Mountain View",
   "types" : [ "locality", "political" ]
   },
   
   { 
   "long_name" : "Santa Clara County",
   "short_name" : "Santa Clara County",
   "types" : [ "administrative_area_level_2", "political" ]
   },
   
   { 
   "long_name" : "California",
   "short_name" : "CA",
   "types" : [ "administrative_area_level_1", "political" ]
   },
   
   { 
   "long_name" : "United States",
   "short_name" : "US",
   "types" : [ "country", "political" ]
   },
   
   { 
   "long_name" : "94043",
   "short_name" : "94043",
   "types" : [ "postal_code" ]
   }
   
   ]
},

CSCI 0931 - Intro. to Comp. for the Humanities and Social Sciences
1) From place to latitude/longitude

- response = # response to a JSON request
- If response["status"] is not "OK"
  - Tell user "Nothing here"
- Else
  - Latitude is in response["results"]
    - which is a list...
  - At position 0 (the most likely response)
    - which is a dictionary
  - At key "geometry"
    - which is another dictionary
  - At key "location"
    - which is yet another dictionary
  - At keys "lat" or "lng"
    - which are numbers!
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)  
    string -> [float,float]'''

    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?
query = placeName.encode('utf-8')
params = {'address': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
json_response = urllib.urlopen(url)
response = json.loads(json_response.read())

    if response['status'] != 'OK':
        return [0.00, 0.00]
else:
    location = response['results'][0]['geometry']['location']
latitude, longitude = location['lat'], location['lng']
return [latitude, longitude]
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude) string --> [float,float]'''
    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else:
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, longitude)
    string -> [float, float]'''

    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else:
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```python
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude) string -> [float,float]'''

googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
query = placeName.encode('utf-8')
params = {'address': query}

url = googleGeocodeUrl + urllib.urlencode(params)
json_response = urllib.urlopen(url)
response = json.loads(json_response.read())

if response['status'] != 'OK':
    return [0.00, 0.00]
else:
    location = response['results'][0]['geometry']['location']
    latitude, longitude = location['lat'], location['lng']
    return [latitude, longitude]
```

The high-level dictionary must have status “OK”, or we had no response
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude) 
    string -> [float,float]'''

    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else:
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
2) From latitude/longitude to place

```python
def printCountry(latlonString):
googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
query = latlonString.encode('utf-8')
params = {'latlng': query}

url = googleGeocodeUrl + urllib.urlencode(params)
json_response = urllib.urlopen(url)
response = json.loads(json_response.read())

if response['status'] != 'OK':
    print 'Nothing found'
else:
    descriptions = response['results'][0]['address_components']

    for description in descriptions:
        if 'country' in description['types']:
            print description['long_name']
```

2) From latitude/longitude to place

```python
def printCountry(latlonString):
    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        print 'Nothing found'
    else:
        descriptions = response['results'][0]['address_components']

        for description in descriptions:
            if 'country' in description['types']:
                print description['long_name']
```

The high-level dictionary must have status “OK”, or we had no response
2) From latitude/longitude to place

```python
def printCountry(latlonString):
    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        print 'Nothing found'
    else:
        descriptions = response['results'][0][
                                  'address_components']

        for description in descriptions:
            if 'country' in description['types']:
                print description['long_name']
```

Get “descriptions”, which is a list of dictionaries (refer to the refer structure)
2) From latitude/longitude to place

```python
def printCountry(latlonString):
    googleGeocodeUrl = 'http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.urlencode(params)
    json_response = urllib.urlopen(url)
    response = json.loads(json_response.read())

    if response['status'] != 'OK':
        print 'Nothing found'
    else:
        descriptions = response['results'][0]['address_components']

        for description in descriptions:
            if 'country' in description['types']:
                print description['long_name']
```

For each dictionary, check if “types” has the kind of descriptor you’re looking for.
You can do much more than print countries!

- **Change** `printCountry` to:
  - Print country, state, and city, if found.
  - Print “Nowhere that I know of otherwise.”

*Hint: refer back to slide 14*

**In groups, do Task 3**
API Details (with much more options)

https://developers.google.com/maps/documentation/geocoding/