Territorial v 1.0

Hernán Morales hernan.morales@gmail.com*

06/09/2016

Contents

*mailto:hernan.morales@gmail.com

Introduction

Territorial[1] is a Smalltalk library for Geographical Information Retrieval (GIR) including features to access geopolitical objects like Nations, Cities, Regions, International Organizations, and statistical data. It could have practical uses in geographical information systems (GIS), game industry to simulate worlds, political sciences and military planning, bioinformatics for determining patient demographics and phylogeography workflows, and many other areas. Special effort was made to enable self-updateable properties when possible, by allowing to query endpoints on-demand, or easy download of remote resources. Another key feature relates to the difficulty of finding good sources of geographical data, for such reason, the library provides a mechanism for transparent interchange of data providers.

The library can be divided into two groups, providing distinctive features: One for working with the existing current and historical world and its main geopolitical views, and another one for defining territories on-demand.

Territorial provides roughly the following features:

- Uniform access to well-known gazetteers and data sets (GeoNames, OpenGeoCode, DBPedia, GADM, ISO-3166, UN.M49, etc.)
- Geographic Information Retrieval : Extract countries, regions, and cities from non-structured text.
- Toponym and exonym resolution for countries and cities.
- Browse available territorial organizations.
- Building historical, fictional, political territories, including a UI wizard.
- Standardized procedures to import new geographical/political data-sets.

Territorial does **not** include:

- Methods for Computational Geometry or Spatial Analysis.
- Operation Research behaviors.
- Bindings to third-party libraries such as GDAL, JTS, GEOS, etc.

[1] http://www.smalltalkhub.com/#!/~hernan/Territorial

It is developed initially for Pharo Smalltalk[2]. The Spec[3] UI library is used in the Territory Browser, and Roassal[4], an open-source visualization engine for Smalltalk is used to render maps. It uses image-based persistency to persist objects between sessions, although custom territories can be persisted through UI using FUEL, a serializer initially developed for Pharo.

Notes and Disclaimers

Territorial is free of charge and far from perfect. The developer does not provide any guarantees for quality, availability, or fitness for particular purpose. Territorial is licensed with the MIT License.

If you feel that you can contribute to **Territorial** development, please do not hesitate to comment or contact the main developer (Hernán Morales) providing suggestions.

[2]http://pharo.org/
[3]http://spec.st
[4]http://agilevisualization.com/

Installation

There are several ways to install the Territorial library. At minimum, you need a working Pharo virtual image installed in your system. Check the Pharo website for installation information regarding the Pharo Open-Source system. Once Pharo is launched you have the following installation options:

Install from Pharo Catalog or Configuration Browser

The easiest way to install the current stable **Territorial** is to use the Pharo Catalog tool. To do it, click in any clean area of the Pharo image and from the ”World menu” open the Pharo Catalog. Search for **Territorial**, select it and click Install.

Install from bash-like command line

You can also install the current stable version from command line in a bash-like shell

```
pharo Pharo.image config \
 ”http://smalltalkhub.com/mc/hernan/Territorial” \
 ”ConfigurationOfTerritorial” --printVersion --install=stable
```

To install a development version, just replace the stable keyword as follows:

```
pharo Pharo.image config \
 ”http://smalltalkhub.com/mc/hernan/Territorial” \
 ”ConfigurationOfTerritorial” --printVersion --install=development
```

Install from a Workspace or Playground

To install the stable package version, copy and paste the following expression into your Pharo image, then select it and perform ”do It” or ”print It”:

```
Gofer it
 smalltalkhubUser: 'hernan' project: 'Territorial';
```

```
 configuration;
 loadStable.
```

To install the development package version, copy and paste the following expression into your Pharo image, then select it and perform ”do It” or ”print It”:

```
Gofer it
 smalltalkhubUser: 'hernan' project: 'Territorial';
 configuration;
 loadDevelopment.
```

Note: If a Warning dialog appears, some external dependencies of Territorial could be failing in some way, please hit Proceed to continue

Conceptual World: Territorial Organizations

In the Territorial model, the World is organized differently according to who, where and when you are. This means there are defined "common" organization objects, and you could decide which one fits better in your application domain. The library is designed to be as agnostic as possible about selecting geopolitical worldviews. Organizations could be actual physical entities (think of UN, IOC, Greenpeace) or any form of organization such as the ones exposed in standard documents (think of ISO 3166-2, ISO 3166-3) officially released by an Institution with the goal of organizing or grouping territories.

Accessing Organizations

Accessing Country Organizations

The following one-liner lists the names of already included country organizations:

```
TCountryOrganization names.
" --> #('EU' 'CARICOM' 'CARIFORUM' 'GADM' 'CCASG' 'ISO 3166-3' '
   Spanish West Indies' 'NATO' 'CEEAC' 'GAUL' 'MERCOSUR' 'Non-Spanish
    West Indies' 'IOC' 'WMO' 'CAEU' 'FIFA' 'UN.M49' 'ISO 3166-2')"
```

Obtaining the currently selected repository for accessing country information is done as follows:

```
TCountryOrganization currentOrganization.
" --> a TCountryOrganization #('ISO 3166-3' 'International ...
   Territories: 241"
```

Country organizations essentially group countries, but they also can be viewed as military alliances, economic communities, standard organizations, etc. For such reason, organizations can be configured with a predefined set of "qualifiers". The following qualifiers are accepted:

- **beGlobal**: For Intercontinental and/or global membership organizations: UN, FIFA, IOC, etc.
- **beEconomical**: For entities involved in economical processes: NAFTA, MERCOSUR, EU, EEA, PIF, ASEAN, etc.
- **beHistorical**: Applies to inactive or dissolved organizations, for example League of Nations, WWII Allies, Communist International.
- **beMilitary**: For military alliances: TAKM, NATO, ANZUS, UNSC, etc.
- **bePolitical**: For entities involved in political processes: UNICEF, Rotary international, LCI, etc.
- **beReligious**: For religious orders, fraternities and secular institutes.
- **beRegional**: For organizations with membership restricted to a particular continent or sub-continental region: Commonwealth of Nations, Shanghai Cooperation Organization, Organization of American States, OPEC, etc.

Organization instances can be queried for its qualifier (#isEconomical, #isGlobal, #isHistorical, #isMilitary, #isPolitical, #isRegional), its period (formed/dissolved date or year), its URL, denomination, etc. Follows some usage scenarios:

Query the formation year of European Union:

```
(TCountryOrganization @ 'EU') formedYear " --> a Year (1957)"
```

Query the denomination of entities:

```
(TCountryOrganization @ 'GAUL') denomination.
" --> 'Global Administrative Unit Layers' "

(TCountryOrganization @ 'CAEU') denomination.
" --> 'Council of Arab Economic Unity'"

(TCountryOrganization @ 'GADM') denomination.
" --> 'Database of Global Administrative Areas'"
```

Probably the most useful feature is requesting member countries from an organization. As the following example to get the members of the *Caribbean Forum*:

```
(TCountryOrganization @ 'CARIFORUM') countries " --> (a
   TerritorialCountry (Saint Lucia) a TerritorialCountry (Dominica) a
   TerritorialCountry (Antigua and Barbuda) a TerritorialCountry (
   Belize) a TerritorialCountry (Trinidad and Tobago) a
   TerritorialCountry (Barbados) a TerritorialCountry (Jamaica) a
   TerritorialCountry (Bahamas) a TerritorialCountry (Grenada) a
   TerritorialCountry (Dominican Republic) a TerritorialCountry (Haiti
   ) a TerritorialCountry (Guyana) a TerritorialCountry (Saint Kitts
   and Nevis) a TerritorialCountry (Saint Vincent and the Grenadines)
   a TerritorialCountry (Suriname) )"
```

To get just the names of countries, replace countries by countryNames:

```
(TCountryOrganization @ 'CARIFORUM') countryNames " --> a Set('Saint
   Lucia' 'Dominica' 'Antigua and Barbuda' 'Belize' 'Suriname' '
   Barbados' 'Jamaica' 'Bahamas' 'Grenada' 'Dominican Republic' 'Haiti
   ' 'Trinidad and Tobago' 'Guyana' 'Saint Kitts and Nevis' 'Saint
   Vincent and the Grenadines') "
```

Requesting a country from different organizations, like ISO[5] or United Nations[6], will answer a **TerritorialCountry object** if registered within that particular organization. A reason behind such design decision is countries have limited recognition[7], and recognition of particular territories changes over time. Requesting the same country from different organizations should answer the same object because **Territorial** identities are resolved by name. The following comparison results in true:

```
((TCountryOrganization @ 'ISO 3166-3') @ 'Argentina') =
 ((TCountryOrganization @ 'UN.M49') @ 'Argentina'). " --> true "
```

[5]http://www.iso.org
[6]http://www.un.org
[7]http://en.wikipedia.org/wiki/List_of_states_with_limited_recognition

If you are working in the context of a particular application, like the FIFA World Soccer Cup, you would want to have the available official registered territories:

```
(TCountryOrganization @ 'FIFA') @ 'China'.
```

Accessing Region Organizations

Regions, in the context of the Territorial library, are territories which could or not be considered as formal geopolitical groupings, for example, the Caribbean which is sometimes considered a sub-region of North America, or a distinct region by the UN.M49. Accessing region organization names is done in similar way to country-based organizations. Territorial already includes some organizations grouped as "Regions":

```
TRegionOrganization names. " --> #('Caribbean' 'Nueva España' '
   Republic of New Granada' 'Rio de La Plata') "
```

For regions there is no default current organization, instead, you should select (or add) your default organization which best fits your purposes.

```
TRegionOrganization currentOrganization: 'Caribbean'.
TRegionOrganization currentOrganization regionNames.  " --> a Set('
   Jamaica' 'Antigua & Barbuda' 'Puerto Rico' 'Montserrat' 'Bahamas' '
   Tortola' 'Bonaire' 'Haiti' 'Dominican Republic' 'Trinidad and
   Tobago' 'Saint Lucia' 'Cayman Islands' 'Virgin Gorda' 'Barbados' '
   Saint Thomas' 'Water Island' 'Marie-Galante' 'Saint Croix' 'Saint
   Kitts' 'Cuba' 'La Désirade' 'Jost Van Dyke' 'Grenada' 'Saint
   Vincent and the Grenadines' 'Aruba' 'Curaçao' 'Anegada' 'Nevis' '
   Turks and Caicos Islands' 'Saint John' 'Anguilla' 'Les Saintes')"
```

Accessing City Organizations

Similarly to countries and regions, accessing city organization names can be obtained in the same way. To list available city organizations:

```
TCityOrganization names. " --> #('DBPedia' 'C40' 'OpenGeoCode')"
```

Obtaining the currently selected repository for accessing city objects is done as follows:

```
TCityOrganization currentOrganization.
```

If the OpenGeoCode repository is the currently used (default), the following two expressions are equivalent:

```
TCityOrganization currentOrganization cities.
(TCityOrganization @ 'OpenGeoCode') cities.
```

Creating Organizations

Creating Country Organizations

The library tries to provide objects for major international organizations. However, it is not feasible to model every possible organization both for the currently available computational and human resources. The Yearbook of International Organizations provided by the *Union of International Associations* lists more than 68,000 organizations, being many of them not currently active. This is why organizations can be created by instantiating commonly recognized characteristics of governmental and non-governmental organizations.

For example adding the G7[8] could be implemented in this way:

```
addG7
 " Answer a new country organization for the G7 "

 ^ TCountryOrganization new
 preferredName: 'G7';
 formedYear: 1975;
 denomination: 'Group of 7';
 bePolitical;
 beEconomical;
 populateWith: 'Canada
France
Germany
Italy
Japan
United Kingdom
United States';
 yourself
```

To view how other organizations are created, you could have a look at the **TCountryOrganization class** (side) methods in category 'territorial-

[8] http://europa.eu/

organizations'.

Real World: Countries, Regions and Cities

Countries

Listing Countries

The UN.M49 is, probably, the most used resource for accessing basic information of UN recognized countries. The UN.M49 is not an institutional organization, but a standard for area codes maintained by the *United Nations Statistics Division* (UNSD). Country lists are an important feature used by many applications, especially dealing with forms (during end-user registration) or search filtering. The following messages are available:

```
| unm49 |
unm49 := TCountryOrganization @ 'UN.M49'.
unm49 sortedTerritoryNames. " ==> a SortedCollection('Afghanistan' '
   Albania' 'Algeria' 'American Samoa' 'Andorra' 'Angola' ... 'Zambia'
    'Zimbabwe' 'Åland Islands')"
unm49 territoryNames.  " ==> a Set('Sierra Leone' 'Guadeloupe' 'Niger
   ' 'Jordan' 'Aruba' ... 'Saudi Arabia' 'Austria' 'Vanuatu' 'Denmark'
    'Lao People''s Democratic Republic')"
```

Countries could be displayed in a friendly-way using a list selector:

```
TerritoryUIList open
    title: 'United Nations Countries';
    roots: (TCountryOrganization @ 'UN.M49') sortedTerritories.
```

Figure 1: United Nations Countries

Accessing Countries

The UN.M49 is the default organization chosen to resolve Country objects using a shortcut with any String matching a country name, for example:

```
'Bulgaria' asTerritorialCountry.
```

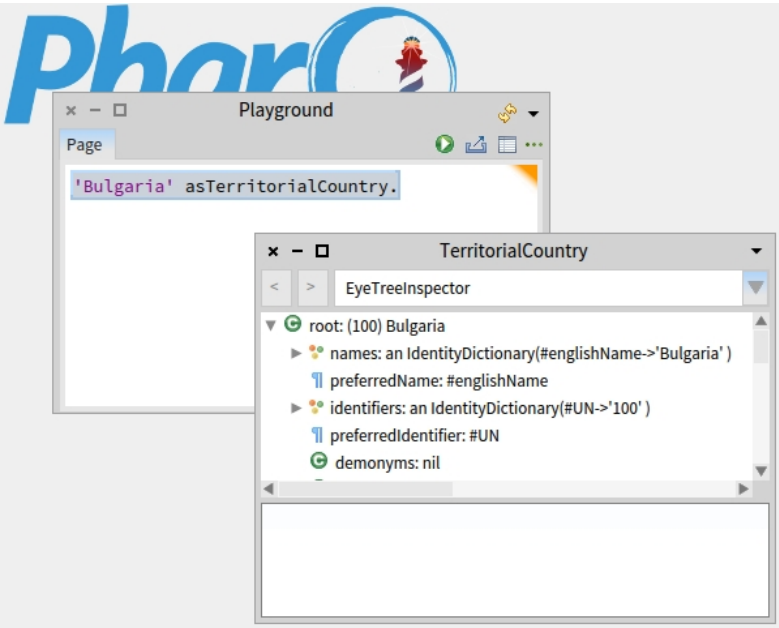

Figure 2: Inspecting Bulgaria as an United Nations country

The country resolver can be changed to any of the available country organizations (see *Accessing Country Organizations*), and can be easily done by evaluating:

```
TCountryOrganization currentOrganization: 'ISO 3166-3'.
```

Evaluating the expression again reflects the territorial object retrieved through the new selected repository :

```
'Bulgaria' asTerritorialCountry.
" Reset to default "
TCountryOrganization currentOrganization: 'UN.M49'.
```

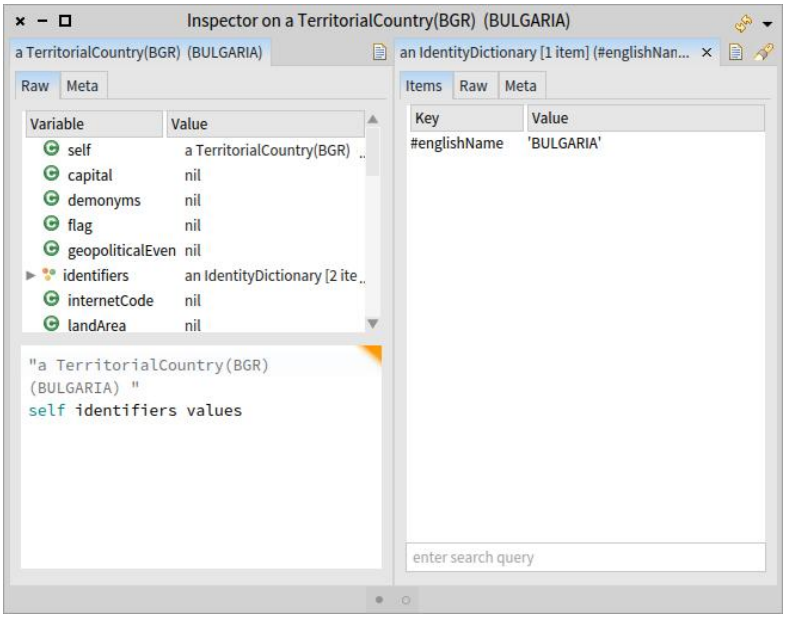

Figure 3: Inspecting Bulgaria as an ISO 3166-3 country

If the country is not registered in the selected organization, the resolver will raise an Error exception. For example, selecting the *Council of Arab Economic Unity* as source organization for countries and then searching for a non-member country :

```
TCountryOrganization currentOrganization: 'CAEU'.
'Ukraine' asTerritorialCountry. " --> Error: Territory not found "
```

It is important to note that some global repositories can be considered as interchangeable, and Territorial can automatically scan all available loaded organizations until a territory name can be found. This behavior can be enabled through the following setting (disabled by default):

```
TCountryOrganization setScanAllOrganizations.
```

An example where such setting can be useful is trying to resolve a country like Taiwan which is not recognized by the UN.M49, but it is found in the IOC repository (and others) through its synonyms: Formosa, Republic of China or Nationalist China :

```
TCountryOrganization isScanningAllOrganizations. " false "
'Taiwan' asTerritorialCountry. " Will raise Not Found exception "
TCountryOrganization setScanAllOrganizations.
'Taiwan' asTerritorialCountry. " a TerritorialCountry(TPE) (Taiwan) "
```

Creating Countries

In the examples for accessing countries, the String protocol was extended to support easy accessing of **existing** countries under some authority. To create a country object, one should use the **TerritorialCountry** class methods. The key difference is that using the String protocol #asTerritorialCountry scans the current organization (or all of them if a preference is set) for an existing country, while **TerritorialCountry** can instantiate any provided name as a country.

```
TerritorialCountry newNamed: 'FakeCountry'.
```

Creating countries could be applicable when building game administrative divisions, for representing historical knowledge, or when working with specific cases like Micronations, which are entities claiming to be independent nations, but commonly are not recognized by major international organizations. In such cases, entities are not accessible through an existing **TCountryOrganization** object (grouping is not mandatory), but nothing prevents from configuring a micronation with Smalltalk messages.

```
(TerritorialComposite newNamed: 'Grand Duchy of Westarctica')
 addName:: 'Communist Republic of Westarctica';
 url: 'http://www.westarctica.info/';
 yourself.
```

Note: Other Micronations exists at the time of this writing: Sealand, Republic of Molossia, Republic of Minerva and Liberland. Note they can issue coins, flags, postage stamps, passports, medals, etc. but such data is not supported in the current version of Territorial, because of most datasets do not recognize such territories

Cities

Cities are first-class entities and can be accessed by name. A city object is a location defined as a permanent or temporary community in which humans live or have lived, without being specific as to size, population or importance. It represents an urban area with autonomous power or a large settlement. A city usually:

- Contains the most of the important administrative offices.
- Merge with, or incorporate surrounding areas.
- Is larger than a town and seat, and more densely populated.

Listing Cities

To view how many cities are in the currently selected repository you can use the #citiesCount selector. The following snippet illustrates querying how many cities are in two repositories :

```
" Query the currently selected organization name "
TCityOrganization currentOrganization denomination. "'Americas Open
    Geocode (AOG) database'"

" Now obtain the count of cities "
TCityOrganization citiesCount. "273034"

" Change the current organization for cities "
TerritorialCities currentRepositoryClass: TerritorialDBPediaCities.

" Get the count of cities for the new organization "
TCityOrganization citiesCount. "9995"

" Revert to the default organization "
TerritorialCities useDefaultRepository
```

To get a list of sorted city names :

```
TCityOrganization cityNames.
```

Note: Only English names are supported in the current version.

Listing Capitals

Capital listings present differences depending of which repository is the currently selected. The current version of the library includes 3 implementations :

Table 1: Country Capitals Providers

Class	Centroid	Count	Includes Country
TerritorialNativeCountryCapitals	No	262	No
TerritorialDBPediaCountryCapitals	No	344	No
TerritorialOpenGeoCodesCountryCapitals	Yes	217	Yes

To access the listings under the implementations above :

```
TerritorialNativeCountryCapitals territorialCapitals.
TerritorialDBPediaCountryCapitals territorialCapitals.
TerritorialOpenGeoCodesCountryCapitals territorialCapitals.
```

Note: Only English names are supported in the current version.

Accessing Cities

In a perfect world, the following expression would be the most intuitive way to name a city:

```
'Ntankuro' asTerritorialCity.
```

However it presents several difficulties:

- It doesn't make sense for cities like "London". Is London the "London, Ontario"? "London, Arkansas"? or "London, Kentucky"?
- Common names could be different from official names. i.e. "Buenos Aires" is used informally, but the official name is "Ciudad Autónoma de Buenos Aires". Such external naming is subject to exonym resolution, for example, Auschwitz is the German exonym for the town of Oświęcim, Londres is Spanish and French for London; Mosca is the Italian name for Moscow, etc. (the United Nations recommends minimizing the use of exonyms in international usage).
- City names can also vary depending accents, availability of character sets available, and how they were mapped in the original data source.

Another source of problems is the repository policy, bugs or interpretation of data. Using New York City as example, in the following expression the first line should raise an exception (since New York is the name of the State), and the second line should answer a TerritorialCity:

```
'New York' asTerritorialCity.
'New York City' asTerritorialCity.
```

But this is not the case in all data sets: The OpenGeoCode dataset has mapped "New York City" as "New York", even when there are several New York locations around the world ("New York, North Yorkshire"; "New York, Lincolnshire"). Considering such sources of ambiguity, the library provides several ways to disambiguate city names.

Disambiguating city providing country name

The next example first configures a country resolver class for cities. It is guaranteed the class ISO3166P1 is already available in the core Territorial library.

```
" Setting the resolver "
TCityOrganization countryResolver: ISO3166P1.
TCityOrganization countryResolveSelector: #atCountryName:.

" Accessing cities "
'Paris@France' asTerritorialCity.
'London@United Kingdom of Great Britain and Northern Ireland'
   asTerritorialCity.
'Alberta@Canada' asTerritorialCity.
```

Disambiguating city providing country code

Let's look at the next example providing an ISO 3166 two letter country code:

```
" Setting the resolver "
TCityOrganization countryResolver: ISO3166P1.
TCityOrganization countryResolveSelector: #at2LetterCode:.

" Accessing cities "
'Paris@FR' asTerritorialCity.
'London@BR' asTerritorialCity.
'Alberta@CA' asTerritorialCity.
```

To help disambiguation, city centroid is also resolved as a **Point** object. All other city properties are obtained through lazy initialization (dynamically on its first accessing). The above implications does not affect all cities, Lille (at France) can be retrieved without country qualifier with city centroid 50.583333@3.083333, which seems to be pretty accurate:

```
'Lille' asTerritorialCity.
```

Issues resolving city names

Always check the current resolver and accessor previous to disambiguate city names:

```
TCityOrganization currentOrganization countryResolver. "ISO3166P1"
TCityOrganization currentOrganization countryResolveSelector. "
   #at2LetterCode: "
```

The above results indicate that a two-letter ISO 3166-1 could be used to successfully access cities after an @ symbol. Notice also a more complete provider could bring problems due to duplicate or non-curated entries:

```
TCityOrganization currentOrganization: 'Native Hand-Crafted'.
'Montevideo@UY' asTerritorialCity isAmbiguousCity. " false "
TCityOrganization currentOrganization: 'OpenGeoCode'.
'Montevideo@UY' asTerritorialCity isAmbiguousCity. " true "
```

Mega-cities

Currently, there is no class to represent a mega-city. The reason is because classifications of mega-cities depend on the number of people (in millions) which contains, and sometimes mega-cities are defined as urban agglomerations instead of metropolitan areas, with an N changing over time. However, a City contains accessors to change its status:

```
'Tokyo@JP' asTerritorialCity beMegaCity.
'Tokyo@JP' asTerritorialCity isMegaCity. "true"
```

There are data sources which contains lists of cities to be considered as mega-cities:

- Demographia[9]
- WikiData[10]

[9] http://www.demographia.com/
[10] https://en.wikipedia.org/wiki/Megacity

- United Nations[11]

Capitals

Capitals are also represented using a **TerritorialCity** object. However, capital status from city organizations can be retrieved from external data sources. Querying and accessing a country Capital city object is possible from a **TerritorialCountry** in the following way:

```
TCountryOrganization currentOrganization: 'UN.M49'.

'Honduras' asTerritorialCountry capital.
" a TerritorialCity (Tegucigalpa) "

'Kyrgyzstan' asTerritorialCountry capital.
"a TerritorialCity (Bishkek) "

'Tunisia' asTerritorialCountry capitalName.
"'Tunis'"
```

One should note that not all countries have capitals, for example, Switzerland where the government is located currently in Bern, but the city is not *De jure* capital. For such cases, an "Unknown" **TerritorialCity** is answered. For countries with multiple capitals (Malaysia, Honduras, South Africa, etc.), the capital method answers the administrative capital for most cases (an exceptional case is Netherlands where the administrative centre is The Hague).

To query if a city is a capital of a country:

```
TCityOrganization currentOrganization. "a TCityOrganization #('
   OpenGeoCode' 'Americas Open... Territories: 273034"
'Pretoria@ZA' asTerritorialCity isCountryCapital. "true"

TCityOrganization currentOrganization: 'Native Hand-Crafted'.
```

[11] https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Highlights.pdf

```
'Athens@GR' asTerritorialCity isCountryCapital. "true"
```

Slums

Note: This section is under development.

Slums, also known as low neighborhood, poor districts or suburbs, can be instantiated.

Data Providers

Territorial includes behavior for updating most of its raw data collections from sources anytime. Such raw collections are accessible in subclasses of **TerritorialDataProvider** class. Wherever possible, freely accessible, cross-domain and usable knowledge graphs in remote endpoints are queried through a SPARQL wrapper[12]). Knowledge graphs not open like Google Knowledge Graph, Google Knowledge Vault, and the Facebook Graph were excluded from the selected datasets. Although Territorial is not intended to be an aggregator or ecosystem of gazetteers, a side consequence of using open geographical data sets could be the augmentation of data quality.

The following table displays the default configuration used in the current version. Reasonable defaults were selected based on the count of names provided by each repository:

Feature	Provider	Method
Translations	FAO	XML
Flags	Free Country Flags	Website
Demonyms	DB1	Internal
Synonyms	WordNet	Python NLTK Corpus
Neighbours	Factbook	SPARQL
Populations	DBPedia	SPARQL
Population Densities	DBPedia	SPARQL
Telephony Country Codes	FreeBase	Website
Telephony City Codes	AGG	Website

It is worth to mention that Data Providers should not be directly referenced by client code. All data providers are used through Organizations (subclasses of **TOrganization**) and Territorial objects (subclasses of **TerritorialObject**). Not all data is contained in Data Providers; for example, most country organizations are populated in TCountryOrganization and therefore not using any Data Provider

[12]http://www.smalltalkhub.com/#!/~hernan/SPARQL

subclasses.

GeoNames

GeoNames is a geographical database that contains more than 8 million geographical names and 7 million unique features. The data is accessible freely through web services and as a downloadable database, updated on a daily basis. GeoNames integrates geographical data such as names of places in various languages, elevation, population and more than 650 territory types from various sources around the world. The size of GeoNames database makes prohibitively expensive for inclusion in the package and should be downloaded (or make available) after installation.

To get a list of all GeoNames feature codes:

```
TerritorialGeoNamesProvider featureCodes
```

GeoNames feature codes can be searched by common (english) name - Peaks, Bays, Lakes, Volcanos, etc.

```
TerritorialGeoNamesProvider detectFeatureNamed: 'peak'. " PK (peak) "
TerritorialGeoNamesProvider detectFeatureNamed: 'lake'. " CHNL (lake
    channel(s)) "
```

And a version for searching feature codes and its description:

```
(TerritorialGeoNamesProvider detectFeatureCodeNamed: 'CNLB')
    geoNameFeatureDescription. " --> 'canal bend' "

(TerritorialGeoNamesProvider detectFeatureCodeNamed: 'RESP')
    geoNameFeatureExplanation. " 'an area of palm trees where use is
    controlled' "
```

To list all available features codes matching a natural language descriptor :

```
TerritorialGeoNamesProvider selectFeaturesCodeNamed: 'lake'  " --> an
    Array(CHNL (lake channel(s)) LBED (lake bed(s)) LK (lake) LKC (
   crater lake) LKI (intermittent lake) LKN (salt lake) LKNI (
   intermittent salt lake) LKO (oxbow lake) LKOI (intermittent oxbow
   lake) LKSB (underground lake) LKX (section of lake) RGNL (lake
   region) ASPH (asphalt lake)) "
```

GeoNames could be accessed through its remote API endpoint. This is useful to debug/compare latest raw contents against its Territorial provider. The following script uses the NeoJSON package, and retrieve/parse the Capitals countries in JSON format:

```
((NeoJSONReader fromString: (
    ZnEasy
      get: 'http://api.geonames.org/countryInfoJSON'
      username: 'demo'
      password: '') contents) at: #geonames) collect: [ : d | d at:
   #capital ]
```

GADM

Note: This section is under development.

GADM is a browseable high-resolution spatial database of world's administrative areas (freely available for academic and other non-commercial use). The data contained in GADM was collected from spatial databases provided by NGO, National Governments, and/or maps and list of names available on the Internet (e.g. from Wikipedia). Administrative areas include countries, provinces, counties, departments, etc. up to five sublevels, which covers most boundaries in the world. For each level, it provides some attributes, foremost being the name and in some cases variant names. GADM can also be used to extract polygon shapes for visualization, for example, to build choropleth maps for regions.

The GADM package can be installed and used independently from Territorial, and includes the raw GADM (version 2) data in CSV format, parsed to build a browseable GADM world tree, allowing off-line access to the GADM database in a hierarchical fashion with objects. This leverages the need to perform on-line queries for basic requests. A hierarchical tree can be used to build a toponym browser for example.

The following examples describes the usage of GADM through the Territorial library:

```
TCountryOrganization currentOrganization: 'GADM'.
'Lithuania' asTerritorialCountry.

GADMWorldTree @ 'Argentina' @ 'Buenos Aires' @ 'La Plata'.
```

Querying Current Data Providers

A Dictionary of current repository mappings can be obtained by evaluating:

```
TerritorialDataProvider currentRepositories.
```

It is useful to check if current providers include a territory when a not found exception is raised. To reset the current data provider for a specific repository, say TerritorialCountries, you can evaluate:

```
TerritorialCountries useDefaultRepository
```

Updating Data Providers

To update a dataset like population densities, evaluate the following expression:

```
TerritorialPopulationDensities release.
```

In this case, it would be useful to keep updated when DBPedia updates census data in its knowledge graph. Then in your application, the population densities will be updated through lazy initialization, without affecting other already populated collections. The next time you query a population density, an SPARQL query to a remote endpoint will populate the local data source.

Territorial Queries

Spellings

There is a partial feature for resolving different spellings to the same entity:

```
(TCountryOrganization @ 'fifa') = (TCountryOrganization @ 'FIFA').
```

However, name resolution is a complex task in itself. To name some typical cases, consider punctuation while naming FIFA as F.I.F.A, or the fact that organizations like the International Olympics Committee are abbreviated differently depending on the language used: Spanish (COI), English (IOC), French (CIO), etc. For such reason, it could take some time to stabilize name resolution feature.

Synonyms

Currently, synonyms are available only for countries. Default synonyms for countries were retrieved using Python NLTK to access WordNet English Corpus, and later manually curated to eliminate inconsistencies.

```
'Poland' asTerritorialCountry synonyms.
" --> #('Poland' 'Republic of Poland' 'Polska') "

'United States' asTerritorialCountry synonyms.
" --> #('United States' 'United States of America' 'America' 'the
   States' 'US' 'U.S.' 'USA' 'U.S.A.') "
```

Demonyms

A demonym, or gentilic, identify residents or natives of a particular place, which is derived from the name of that particular place. Demonyms are used commonly for country-level, state-level (province-level) or city-level territories.

Country Demonyms

To access a list of demonyms for a country, first resolve the country as a **TerritorialCountry** and the request for its #demonyms:

```
'Niger' asTerritorialCountry demonyms. " --> #('nigerien')"
'Nigeria' asTerritorialCountry demonyms. " --> #('nigerian')"
```

As there could be multiple demonyms for a territory, results always will be a Collection of String. Demonyms are important in bioinformatics text-mining systems for recognizing breeds or species names. For such kind of cases, the library supports finding demonyms in arbitrary Smalltalk strings:

```
'Hungarians, also known as Magyars (Hungarian: magyarok), are a nation
    and ethnic group who speak Hungarian and are primarily associated
   with Hungary.' parseCountryDemonym " --> 'hungarian' "
```

City Demonyms

Of course, demonyms also exists for cities. **Territorial** is configured to use by default the DBPedia cities (i.e. class City in the DBPedia ontology). This could mean Cities categorized as Towns (or similar classifications) are missing in the imported data set. To retrieve demonyms for a city one first should resolve it as a **TerritorialCity** object:

```
'Cali@CO' asTerritorialCity demonyms. " -> #('caleño') "
```

Note that DBPedia city names are not curated and in some cases a city is named in the pattern ”name, state” or ”name, country”. If no demonym is found, the following expression could be used to examine the current repository of city demonyms:

```
TerritorialDemonyms currentCityRepository cityDemonyms.
```

We can observe cities as ’Columbia’ and ’Matara’ are named as ’Columbia, Missouri’ and ’Matara, Sri Lanka’ respectively. **Territorial** is not intended to be as a library for curating remote databases, and it is recommended that data curation effort be directly requested to any of the Linked Open Data repositories (as DBPedia, WikiData, and others).

In the case of obtaining multiple cities for a name, a situation which looks like this in the Inspector:

```
'Los Angeles@US' asTerritorialCity
```

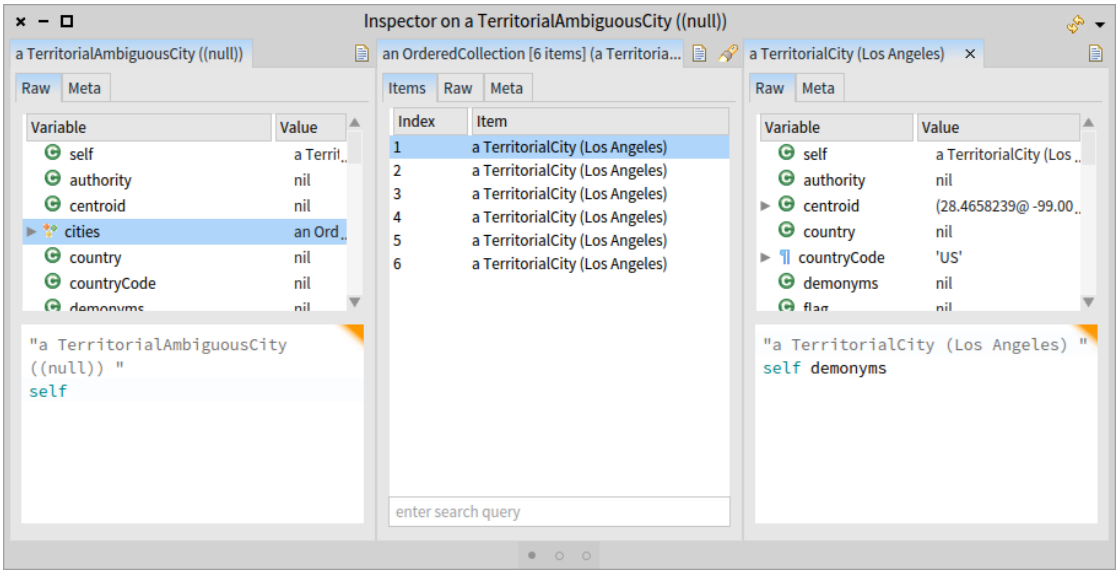

Figure 4

To disambiguate it, you can ask for any location in an Ambiguous City.

```
'Los Angeles@US' asTerritorialCity anyOne demonyms. " -> #('angeleno')
    "
```

Translations

Translated names enable to query an entity name in different languages and it is resolved to the same object. Territorial includes two datasets to provide translations:

- 21,640 country names in UTF-8 CSV file[13] from Jonah Ellison.
- FAO translations from their Geopolitical ontology country profiles[14].

Compare different translations for the same country:

```
'Svizzera' asTerritorialCountry = 'Svizra' asTerritorialCountry.
```

Comparing different countries (commonly confused as the same country):

```
'The Democratic Republic of Congo' asTerritorialCountry ~=
 'Democratic Republic of Congo' asTerritorialCountry.
```

Obtain all the known names (in the current dataset) for France:

```
'France' asTerritorialCountry allNames.

" --> a Set('ffrainc' французская' республика' 'pranses' francie' '
   frankreich' □□□□'' □□□□□□'' 'france' 'repubblica francese' '
   falansia' 'an fhrainc' 'alang posey' γαλλία'' франция'' □□'' □□□□''
    □□□□□□' ฝรั่ง'' франција'' 'páp' 'fransa' 'frakkland' □□□''
   француска'' 'the french republic' 'ufaransa' 'frankriich' 'francë'
   'ţfrana' 'francija' 'frantzia' □□□□□'' франция'' 'frankrig' 'la
   república francesa' 'frakland' 'la république française' '
   ūpranczija' 'la france' □□□□'' 'perancis' 'faransiis' □□□□□''
   □□□□□' □□'' 'frankrijk' 'France' 'palani' 'frankrike' □□□□□''
   □□□□□□' 'prantsusmaa' 'frankriek' □□□□□□'' هسنارف'' سنارف'' 'França
   ' □□□□□□' □□□□□□□' □□□□'' ةيرومجلا' ةيسنرفلا' □□□□□'' франція'' '
   franciaország' 'francujo' 'ēfalanis' 'francja' 'frankriika' 'franza
```

[13] http://jonahellison.com/21640-translated-country-names-unicode-csv
[14] http://www.fao.org/countryprofiles/geoinfo/en/?lang=en

```
    ’ ֆրանսիա’’ □’ □’ франц’’ السنرف’’ ’francia’ □□□□□’’ □□□□□’
    საფრანგეთი’’ ’francúzsko’ □□□□□’’ ’ranska’ □□□□□□□’’ ’francuska’)”
```

Note that accessing names is different from accessing translations, where a mapping of language -> translated name is obtained. To compare both outputs inspect the following expressions separately:

```
’Norway’ asTerritorialCountry translations.
’Norway’ asTerritorialCountry allNames.
```

Telephony Codes

Country Telephony Codes

Obtaining calling codes from countries is similar to previously discussed sections:

```
'India' asTerritorialCountry telephonyCodes. " '91' "
```

To access the dial codes for countries in the currently selected repository:

```
TerritorialTelephonyCodes currentCountryRepository
    countryTelephonyCodes.
```

City Telephony Codes

To make long distance international telephone calls is necessary to dial a telephone number city code. Although the formatting of each result is the responsibility of the data provider used to resolve dialing codes, in most providers evaluated the resulting code could include a leading 0 (zero) number, so called "trunk code", and a plus sign, which means the country exit code.

To get a String with a city calling code, providing a country for disambiguation (See section Disambiguating city providing country code/name) one can evaluate the following expression:

```
'Iwate@JP' asTerritorialCity telephonyCodes. " '(0)195' "
```

If a city cannot be resolved because is not present in the current repository, or is not present in any repository, an alternative way to find out the city code is to query directly the repository class, in the format:

'city_lowercased_name@CountryName'

For example:

```
'alchevsk@UA' asTerritorialCity. " Not Found "
```

```
TerritorialTelephonyCodes currentCityRepository cityTelephonyCodesAt:
    'alchevsk@Ukraine'. "'(8~0)6442'"
```

To access the dial codes for cities in the currently selected repository:

```
TerritorialTelephonyCodes currentCityRepository cityTelephonyCodes.
```

Demographics

Human population statistics are available in Territorial currently by means of the dbpedia database[15] for both census and estimated population data. To see how many humans were registered by census in Austria along with density in square kilometers, evaluate the following code:

```
'Austria' asTerritorialCountry population " --> 8032926 "
'Austria' asTerritorialCountry populationDensity " --> 101.39 "
```

To access the estimated population sizes:

```
'Vietnam' asTerritorialCountry estimatedPopulation " --> 90630000 "
```

The obvious advantage of querying the DBPedia dataset is obtaining automatic updates of census data from the Linked Open Data cloud.

Querying country neighbourhood is also possible with linked data provided by a D2R Server for the CIA Factbook[16].

```
'India' asTerritorialCountry neighbours. " --> #('bangladesh' 'bhutan
   ' 'burma' 'china' 'nepal' 'pakistan') "

'Honduras' asTerritorialCountry neighbours. " --> #('guatemala' 'el
   salvador' 'nicaragua') "
```

[15] http://dbpedia.org
[16] http://wifo5-03.informatik.uni-mannheim.de/factbook/

Flags

Default country flags were taken from Free Country Flags[17] (Creative Commons Attribution-ShareAlike 3.0 Unported License) in three sizes:

- Tiny: 20 x 12 Pixels
- Small: 80 x 39-59 Pixels
- Medium: 236 x 115-175 Pixels

To visualize a flag for example in an ImageMorph[18], you just specify the territory name:

```
TerritorialFlags currentCountryRepository openCountryFlagMediumNamed:
    'gabon'.
TerritorialFlags currentCountryRepository openCountryFlagMediumNamed:
    'mauritania'.
TerritorialFlags currentCountryRepository openCountryFlagMediumNamed:
    'belize'.
```

Figure 5: Displaying country flags in Pharo

[17] http://www.free-country-flags.com
[18] http://wiki.squeak.org/squeak/morphic

Geographical Data

This section describes values such as land area, water area, etc. as currently (2016) returned in different Linked Open Data repositories. The current installation already contains queries for the DBPedia dataset. Other well-known sources, partially integrated or with planned integration in Territorial, are: CIA World Factbook, WorldBank, GeoNames and FAO Geopolitical Ontology.

Geographical data like land area could differ between sources, specially with current problematic regions: Islas Malvinas (Falkland Islands), British Indian Ocean Territory, Guantanamo Bay, Akrotiri and Dhekelia, and/or considering French overseas departments and the municipalities of the Caribbean Netherlands as separate or integral parts of France and the Netherlands, respectively.

Geographical Areas

The land area is a **Number** representing a territory total area in square kilometers (km2), excluding exclusive economic zones, area under inland water bodies and national claims to continental shelf. As an example one can evaluate the following expression to obtain the land area in Km2 of Uruguay :

```
'Uruguay' asTerritorialCountry landArea " -> 176215 "
```

Similarly, the total area is a Number representing a territory total area (land and sea) in square kilometers (km2).

```
'Canada' asTerritorialCountry totalArea " ->  9982.03 "
```

Maps

Note: *Maps in Territorial are currently under heavy development Territorial does not include (yet) features for edition of maps*

Any territorial object which inherits from **TerritorialMappableObject** can have associated any number of maps. Maps are accessed through the protocol "accessing - maps". Internally, each territorial object uses a map provider which provides a Collection of maps (each map is a **TerritorialExternalMap**). Such implementation permits to associate any map from any source, such as a local or remote static URL, and to be rendered through different map services like OpenStreetMaps, GoogleMaps, BingMaps, etc. without losing the map properties or user-defined annotations. Multiple implementations for a particular map service is also supported. A map can have any number of "owners", meaning that multiple territorial objects can share the same map.

Adding Maps and Displaying Maps

Static maps can be added to and retrieved from any Territorial Object or Organization :

```
'Syria' asTerritorialCountry
  addMapUrl: 'http://bit.ly/2aqfZ4Y' description: 'Syria (Wikipedia
    Map)';
  addMapUrl: 'http://bit.ly/2aG5I9q' description: 'Syria (Google Maps)
    ';
  openMaps;
  yourself.
```

OpenStreetMaps

Installation

Due to errors including OSM dependencies in the Territorial Metacello Configuration, both two OSM packages should be installed separately in the image in order to access any dynamic map features:

```
Metacello new
    baseline: 'CirelaOSM';
    repository: 'github://OnilGoubier/Cirela';
    load.
```

```
Metacello new
  baseline: 'OSMMaps';
  repository: 'github://fstephany/OSMMaps/repository';
  load.
```

After installing external OSM support, you can install the Territorial package for OpenStreetMaps:

```
Gofer it
  smalltalkhubUser: 'hernan' project: 'Territorial';
  package: 'TerritorialData-OSM';
  load.
```

Displaying Maps with OpenStreetMaps

The following expression opens a window with Paris (France) map using one of the OpenStreetMap implementations as map provider:

```
TerritorialMaps currentRepository: TerritorialOSM
'Paris@FR' asTerritorialCity openMap.
```

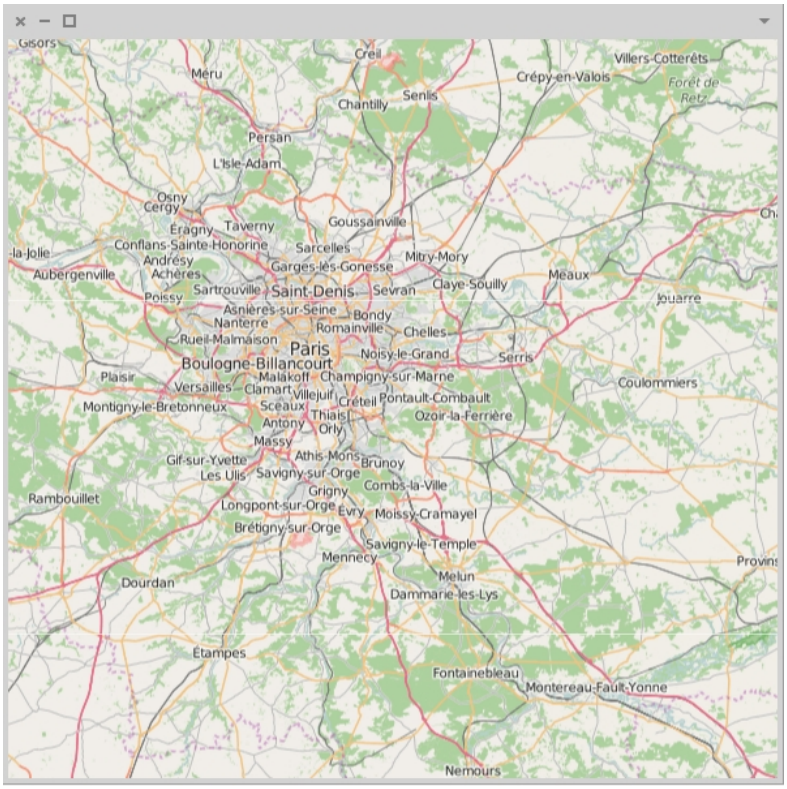

Figure 6: Paris at zoom level = 9

It is possible to change the OSM map provider by setting another OSM class repository:

```
TerritorialMaps currentRepository: TerritorialOSM
'Paris@FR' asTerritorialCity openMap.
```

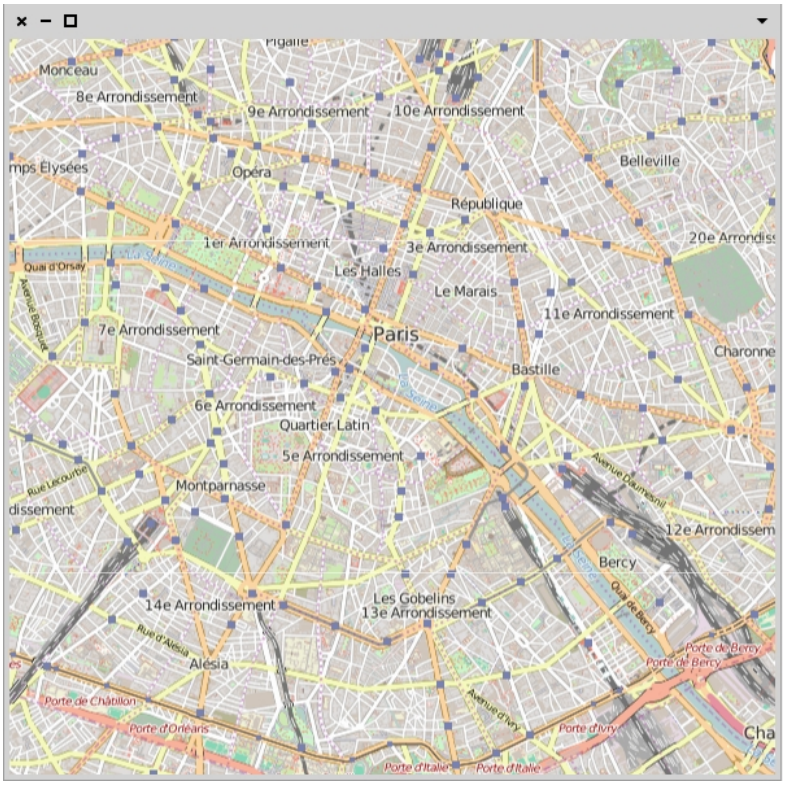

Figure 7: Paris at zoom level = 13

GoogleMaps

Note: Section under development

There are a few implementations of GoogleMaps API in Smalltalk. Two of them were developed for use with the Seaside Web Framework and are not yet isolated to work without the Seaside Framework. The remaining implementation uses the Roassal visualization engine, and can be installed in Pharo with the following expression:

```
Gofer it
    smalltalkhubUser: 'smaass' project: 'GMaps';
    package: 'GMaps';
    load.
```

After installing support for Google Maps, you can install the Territorial package helper:

```
Gofer it
  smalltalkhubUser: 'hernan' project: 'Territorial';
  package: 'TerritorialData-GoogleMaps';
  load.
```

Map Decorations

Note: Section under development

Geopolitics

Support for Geopolitics is implemented currently through three operations applied to Territories: Amalgamation, Colonization, and Conquest.

Amalgamation, Colonization and Conquest

- Amalgamation: Pacific union of two States o geographical spaces sharing the same interests. It is also known as merger or consolidation.
- Colonization: Application of cultural, military, economic and/or scientific influential practices.
- Conquest: Application of forces to obtain subjugation of another State.

Examples

Occupations

Russian aggression in eastern Ukraine

The following example creates an Occupation object. The Russian occupation in Ukraine caused thousands of casualties from both pro-Russian and pro-Ukranian sides, foreign civilians and volunteers. Several source URL's could be added to an occupation, to link with authoritative information, as an on-going conflict many sources could be in dispute.

```
| russia easterUkraine occupation |
russia := TCountryOrganization @ 'Rusia'.
" East of Ukraine is occupied currently, this could be specified later
   ..."
ukraine := 'Ukraine' asTerritorialCountry.
ukraineOccupation := ukraine
 occupedBy: russia
 namedAs: 'Russian – Ukraine 2014 Occupation'
 period: (Date year: 2014 month: 2 day: 20) -> Date today;
```

```
 yourself.
ukraineOccupation.
  addUrl: 'https://en.wikipedia.org/wiki/
   Russian_military_intervention_in_Ukraine_(2014%E2%80%93present)'
   score: 1;
  addUrl: 'http://ukraine.csis.org/' score: 2;
  addUrl: 'http://www.globalsecurity.org' score: 3;
  beMilitary;
  yourself.
```

Occupation of Argentine's Islas Malvinas by the British Empire

The occupation of *Islas Malvinas* by the British Empire is an ongoing conflict between *Argentina* and *United Kingdom of Great Britain and Ireland* (previously the British Empire), originated by British interests that were part of the War of Jenkins' Ear between Spanish Empire and British Empire, from 1739 to 1748. It could be represented as follows:

```
(TerritorialArea newNamed: 'Islas Malvinas')
        occupedBy: 'British Empire' asTerritorialCountry
        namedAs: 'British Empire Occupation of the Islas Malvinas'
        period: (Date year: 1833 month: 1 day: 3) -> Date today.
```

Colonizations

Subdivisions in Lybia

Suppose we wish to represent part of the historical subdivisions in Lybia. What is known today as Lybia has a long history (from 146 B.C.) of reorganizations and occupations. To delimit its history, we are going to take our example of Lybia from the point where it was part of the Ottoman Empire. This is our initial situation:

```
| lybia |
lybia := (TerritorialComposite newNamed: 'Ottoman Empire')
 addTerritory: (TerritorialComposite newNamed: 'Tripolitania');
 addTerritory: (TerritorialComposite newNamed: 'Cyrenaica');
 addTerritory: (TerritorialComposite newNamed: 'Fezzan');
 addPeriod: 1551 asYear -> 1911 asYear
 yourself.
```

After 1911, *Italy*, the *WW2 Allies*, and the *Arab League* were part of the Lybian history by occupation, administrative tasks or colonization. We can translate such geopolitical events as follows:

```
| italy |
allies := TCountryOrganization @ 'WW2 Allies'.
arabLeague := TCountryOrganization @ 'Arab League'.
italy := 'Italy' asTerritorialCountry.

lybia
 occupedBy: italy      namedAs: 'Italian occupation' period: 1911
    asYear -> 1934 asYear;
 colonizedBy: italy    namedAs: 'Italian Libya'      period: 1934
    asYear -> 1943 asYear;
 occupedBy: allies     namedAs: 'Allied occupation'  period: 1942
    asYear -> 1951 asYear;
 occupedBy: arabLeague namedAs: 'Kingdom of Libya'   period: 1951
    asYear -> 1969 asYear;
 yourself.
```

Annexations

Annexation of Tibet by People's Republic of China (PRC)

An example of annexation is the conflict between the old autonomous region *Tibet*, a strategic zone of water sources, and *PRC*. *Tibet* also acts as a Buffer Zone between *PRC* and *India*, and the sovereignty of political *Tibet* is a source of debate.

```
(TerritorialArea newNamed: 'Tibet')
        annexedBy: 'People''s Republic of China' asTerritorialCountry
        namedAs: 'Incorporation of Tibet into the People''s Republic
    of China'
        period: ((Date year: 1951 month: 10 day: 1) -> Date today).
```

Territory Building

Territory Building through code

A Game of Thrones example

In the example below you can see how to build the *Game of Thrones* main territories using the Territorial messages. Game of Thrones[19] administrative regions of the *Seven Kingdoms* consists basically of three "continents". These continents contains political regions. In a second example we are going to configure the same territory but added several random second-level regions to show some available features.

As you may note, I have used the TerritorialComposite class, because the containment could be represented using the Composite Design Pattern[20].

```
'Known World' asTerritorialComposite
    addTerritory: ('Westeros' asTerritorialComposite
        addTerritory: 'Beyond the Wall' asTerritorialComposite;
        addTerritory: 'The North' asTerritorialComposite;
        addTerritory: 'Iron Islands' asTerritorialComposite;
        addTerritory: 'The Vale of Arryn' asTerritorialComposite;
        addTerritory: 'The Riverlands' asTerritorialComposite;
        addTerritory: 'The Westerlands' asTerritorialComposite;
        addTerritory: 'The Reach' asTerritorialComposite;
        addTerritory: 'The Stormlands' asTerritorialComposite;
        addTerritory: 'The Crownlands' asTerritorialComposite;
        addTerritory: 'Dorne' asTerritorialComposite;
        yourself);
    addTerritory: ('Essos' asTerritorialComposite
        addTerritory: 'Dothraki Sea' asTerritorialComposite;
        addTerritory: 'Free Cities' asTerritorialComposite;
        addTerritory: 'Ghiscar' asTerritorialComposite;
```

[19] http://www.hbo.com/game-of-thrones
[20] http://c2.com/cgi/wiki?CompositePattern

```
        addTerritory: 'Ifequevron' asTerritorialComposite;
        addTerritory: 'Jhogwin' asTerritorialComposite;
        addTerritory: 'Jogos Nhai' asTerritorialComposite;
        addTerritory: 'Lhazar' asTerritorialComposite;
        addTerritory: 'Mossovy' asTerritorialComposite;
        addTerritory: 'Red Waste' asTerritorialComposite;
        addTerritory: 'Sarnor' asTerritorialComposite;
        addTerritory: 'Shadow Lands' asTerritorialComposite;
        addTerritory: 'Slaver''s Bay' asTerritorialComposite;
        addTerritory: 'Valyria' asTerritorialComposite;
        addTerritory: 'Yi Ti' asTerritorialComposite;
        yourself);
    addTerritory: ('Sothoryos' asTerritorialComposite
        addTerritory: 'Ax Isle' asTerritorialComposite;
        addTerritory: 'Basilisk Isles' asTerritorialComposite;
        addTerritory: 'Basilisk Point' asTerritorialComposite;
        addTerritory: 'Isle of Tears' asTerritorialComposite;
        addTerritory: 'Isle of Toads' asTerritorialComposite;
        addTerritory: 'Naath' asTerritorialComposite;
        addTerritory: 'Wyvern Point' asTerritorialComposite;
        yourself);
    yourself.
```

The *Vale of Arryn* was known formerly as *The Kingdom of Mountain and Vale*, and so we can add a former territory name using #addFormerName:, another supported feature is adding territories not yet established as cities or towns, as **TerritorialSettlement**. Now see a more complete *Westeros* building:

```
'Known World' asTerritorialComposite
    addTerritory: (
        'Westeros' asTerritorialComposite
            addTerritory: 'Beyond the Wall' asTerritorialComposite;
            addTerritory: 'The North' asTerritorialComposite;
```

```
        addTerritory: ('Iron Islands' asTerritorialComposite
                addDemonym: 'Ironborn';
                yourself);
        addTerritory: ('The Vale of Arryn' asTerritorialComposite
                addFormerName: 'The Kingdom of Mountain and Vale';
                yourself);
        addTerritory: ('The Riverlands' asTerritorialComposite
                url: 'http://gameofthrones.wikia.com/wiki/
The_Riverlands';
                addFormerName: 'Region of the Kingdom of the Isles
 and the Rivers';
                yourself);
        addTerritory: 'The Westerlands' asTerritorialComposite;
        addTerritory: 'The Reach' asTerritorialComposite;
        addTerritory: 'The Stormlands' asTerritorialComposite;
        addTerritory: 'The Crownlands' asTerritorialComposite;
        addTerritory: ('Dorne' asTerritorialComposite
            addDemonym: 'Dornishmen';
            addTerritory: ('Sunspear' asTerritorialComposite
                url: 'http://gameofthrones.wikia.com/wiki/Sunspear
';
                addTerritory: (TerritorialSettlement newNamed: '
Vulture''s Roos');
                yourself);
        yourself)
    )
```

Territory Building through User-Interface

Territorial provides a wizard user-interface for easy territory building. It could be opened by evaluating:

```
TerritoryBuilderUIChooser open.
```

The first window presents two choices:

- Create a new territorial composite or single territory.
- Browse already created territories.

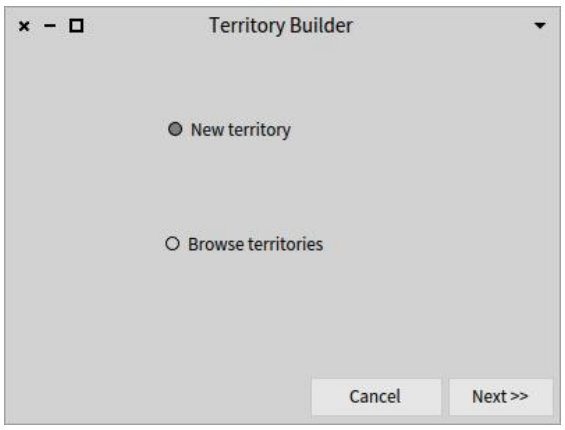

Figure 8: Territorial Wizard

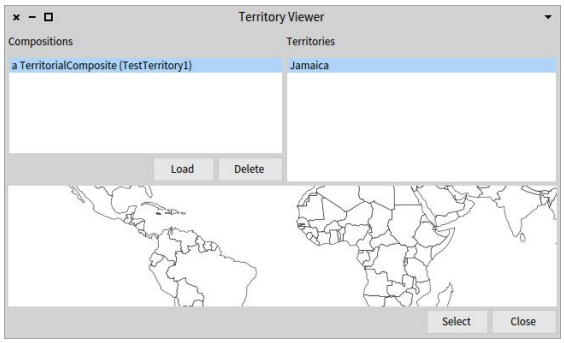

Figure 9: Territory Viewer

Browsing Territories

Browsing presents a new viewer window with in-image territories, and a Load button to load previously saved territories. Territories are saved in FUEL format, a binary serializer developer originally for Pharo. The viewer window is intended to be specialized for particular domains where selection of pre-built territories may be appropriate.

Creating Territories

Composition of territories is sub-divided in two additional choices. Creating a new composite territory or creating/selecting a new single territory.

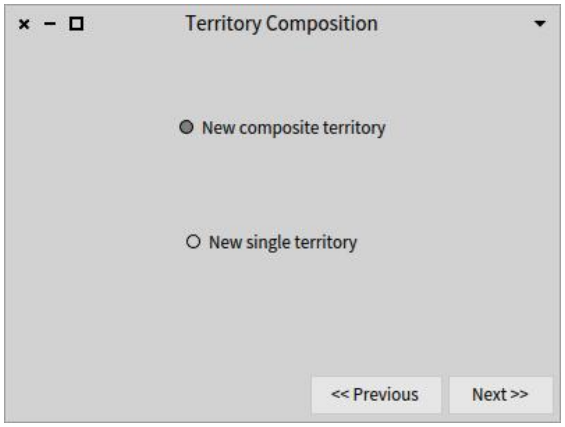

Figure 10: New Composite Territory

Next to selection of a New Territory, the wizard request for a name to identify the new territory:

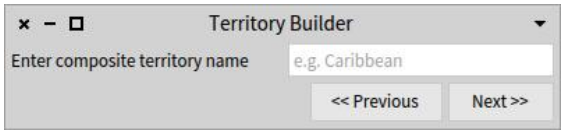

Figure 11: New Composite Territory name

Finally, a Territory Composer is displayed where territories can be browsed and composed (or selected if a single territory was chosen).

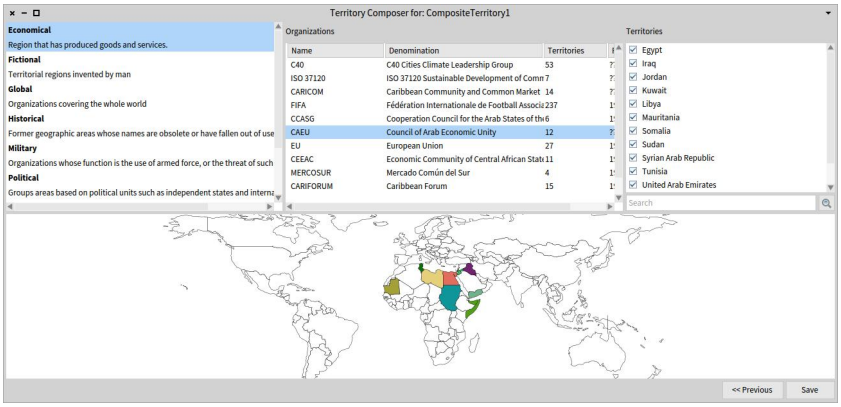

Figure 12: Territory Composer for Composite selection

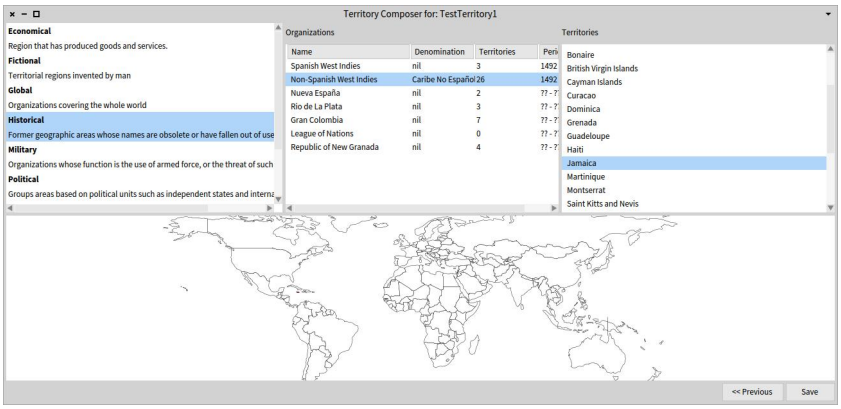

Figure 13: Territory Composer for Single territory creation

Developing Territorial

Future versions

- Add IANA internet codes.
- Add more GeoNames features.
- Add Bioregions.

Contributing Guidelines

- Have a look at existing code to get a feel.
- Indent code using tab, not spaces.
- No unnecessary brackets.
- Close multiple blocks on one line.
- Add a space at the beginning and end of a temporary variable declarations and blocks.
- Don't use any dialect specific syntax, like byte array literals #[ ], curly braces { }, or pragmas .
- Instead of TimeStamp use DateAndTime.
- Do not assume #value is part of the Object protocol.
- Use WriteStream on: String new instead of String new writeStream.
- Don't send #next to a ReadStream that is at end. Instead send #atEnd to check if the stream is at the end.

Reporting Issues

- Issues should be reported at the current ticket system at : https://github.com/hernanmd/Territorial/issues
- Please always include your environment information (Virtual Machine used with version, Image version used, Operating System, Stable or Development versions of Territorial)
- To ask questions please use the ettiquete suggested at http://www.catb.org/~esr/faqs/smart-questions.html

How to Create a Data Provider

If you want to include new data and API into Territorial, you could take the following recommended steps:

- If the data source contains files to be imported (such as CSV, XML, etc.), create a directory (for example RESOURCE_DIRECTORY_NAME) under the Territorial datasets directory (currently named ”territorial_files”) and place the data files into such created directory.
- Create a new package prefixed TerritorialData-??? where ??? is the new package name.
- Locate or create a TerritorialData subclass, that class will be used to access the resource and will provide the importing behavior and raw accessing to the data, for example in the form of OrderedCollection, Dictionary or Set.
- Create a method named #myResourceFilename answering the name of the file containing the resource.
- Create an importer method using the following template:

```
myResourceAccessor
    ” Read a file of source dataset ”

    ^ self readFileContents: self myResourceFilename
```

Troubleshooting

Space is low (out of memory exception)

If you are loading multiple datasets and your image raise a dialog with a **Space is low** condition, there are several workarounds you could try to clean up space in your image:

Flush Monticello caches:

```
MCFileBasedRepository flushAllCaches
```

Remove tests from the image (could take some time)

```
RPackageOrganizer default packageNames
 select: [ : each | each endsWith: 'Tests' ]
 thenDo: [ : each | (MCPackage named: each) unload ].
```

Remove all loaded data sets (then you can re-execute your code with necessary data sets):

```
TerritorialDataProvider release.
```

License