>BULK

Simulation and management program for societies.

Dunstan Becht

Bulk is the second part of the Pale Blueprint initiative. It aims to create a test platform for societies designed on the principles enunciated in Entropy Economy System.[A.1] It is therefore preferable to read the first project before. This document is a non-exhaustive presentation of the aspects and functionalities of the program. It marks the release of the version 0.8.0 and the beginning of the alpha phase in the release life cycle.

Contents

1 A web application

The current objectives in the development of the Bulk are the following functionalities:

- Simulate the economic activity of a population, and therefore energy exchanges and losses [A.1].
- Compare different models of societies according to their organizations and operating rules.
- Allow immersion in the modeled society via an interface that would be that of a citizen.

A web application was seen as the way to meet these different needs. It was essential to make the interface easy to access, which is the case today. The Bulk is hosted at the following URL address:

🌐 `https://bulk.paleblueprint.org`

1.1 Languages

The Bulk is developed with the following languages:

- PHP [A.2.1]
- MySQL [A.2.2]
- HTML5 [A.2.3]
- CSS3 [A.2.3]
- JavaScript [A.2.3]

1.2 Model–view–controller

A MVC architecture [A.3.1] is used. URL addresses are built on the following pattern:

`https://bulk.paleblueprint.org/controller/action/parameters`

It is possible to navigate between 5 controllers:

- `home`
- `interface`
- `overview`
- `information`
- `settings`

1.3 Post-redirect-get

Placed at the root, the file `index.php` sets up the method post-redirect-get: [A.3.2]

index.php

```php
if (count($_POST)>0) {
  foreach ($_POST as $key => $value){
    $_SESSION[$key] = htmlspecialchars($value);
  }
  header("HTTP/1.1 303 See Other");
  header("Location: https://$_SERVER[HTTP_HOST]$_SERVER[REQUEST_URI]");
  die();
}
```

2 Models

The data managed and used by the Bulk mainly consists of the objects introduced in section 3 of the Entropy Economy System report [A.1] $(E_\wedge, S_\wedge, P_\wedge, C_\wedge, T_\wedge, R_\wedge, L_\wedge)$.

2.1 Relational database

The Bulk uses a database containing the following relations. Primary keys are underlined and arrows indicate to which column a foreign key refers.

- `energy` **(3.1.1)**[A.1]
  - `id (INT)`
  - `name (STR)`
- `system` **(3.2.1)**[A.1]
  - `id (INT)`
  - `name (STR)`
  - `password (STR)`
- `position` A.1[A.1]
  - `system→system(id) (INT)`
  - `dimension (INT)`
  - `value (INT)`
- `capacity` A.1[A.1]
  - `system→system(id) (INT)`
  - `energy→energy(id) (INT)`
  - `quantity (INT)`
- `tree` **(3.2.1)**[A.1]
  - `parent→system(id) (INT)`
  - `descendant→system(id) (INT)`
- `pointer` **(3.3.2)**[A.1]
  - `quantity (INT)`
  - `energy→energy(id) (INT)`
  - `storer→system(id) (INT)`
  - `owner→system(id) (INT)`
- `channel` **(3.4.2)**[A.1] and **(3.5.2)**[A.1]
  - `id (INT)`
  - `sender→system(id) (INT)`
  - `receiver→system(id) (INT)`
  - `input→energy(id) (INT)`
  - `output→energy(id) (INT)`
  - `efficiency (FLOAT)`
  - `power (INT)`
  - `delay (INT)`

- `register` **(3.6.2)**[A.1]
  - `id` (INT)
  - `channel→channel(id)`
  - `start (INT)`
  - `end (INT)`
  - `power (INT)`
- `monitor` 4.3[A.1]
  - `id` (INT)
  - `channel→channel(id)`
  - `start (INT)`
  - `end (INT)`
  - `power (INT)`

2.2 Representatives

Data is represented by instances of classes inheriting from the class Representative.

Representative.php

```php
abstract class Representative {

  // Hydrate:
  public function hydrate(array $data) {
    foreach ($data as $key => $value) {
      $method = "set".ucfirst($key);
      if (method_exists($this, $method)) {
        $this->$method($value);
      } else {
        throw new \Exception("No attribute ".$key." in ".static::class);
      }
    }
  }

  // Construct:
  public function __construct(array $data) {
    $this->hydrate($data);
  }

  // Display:
  public function display() {
    $lines = array();
    foreach (get_object_vars($this) as $key => $value) {
      if (is_array($value)) {
        $value = "[".join(", ", $value)."]";
      }
      array_push($lines, ucfirst($key).": ".$value);
    }
    return join("<br>", $lines);
  }
}
```

2.3 Representative managers

The only purpose of the previous class is to represent the data. To manipulate this data, classes inheriting from class RepresentativeManager are used.

RepresentativeManager.php

```php
abstract class RepresentativeManager {

  // Database:
  private $database;
  public function getDatabase() { return $this->database; }
  public function setDatabase(\PDO $value) { $this->database = $value; }

  // Construct:
  public function __construct($database) { $this->setDatabase($database); }

  // Read:
  public function getList(array $columns=array(), bool $like=TRUE) {
    $assertions = array();
    foreach ($columns as $key => $value) {
      if (is_string($value)) {
        if ($like) {
          array_push($assertions, $key." LIKE '%".$value."%'");
        } else {
          array_push($assertions, $key." = '".$value."'");
        }
      } else {
        array_push($assertions, $key." = ".$value);
      }
    }
    if (sizeof($assertions)==0) {
      $condition = "";
    } else {
      $condition = " WHERE ".join(" AND ", $assertions);
    }
    $class = substr(static::class, 0, -7);
    $table = strtolower(substr($class, 7));
    $q = $this->getDatabase()->query('SELECT * FROM '.$table.$condition);
    $data = $q->fetchall(\PDO::FETCH_ASSOC);
    $result = array();
    foreach ($data as $objectData) {
      array_push($result, new $class($objectData));
    }
    return $result;
  }
}
```

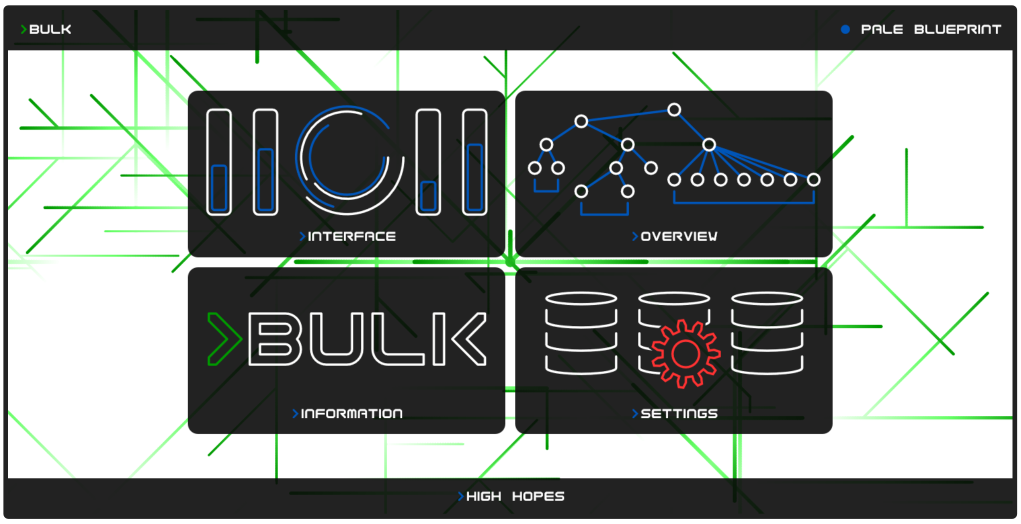
2.4 Societies

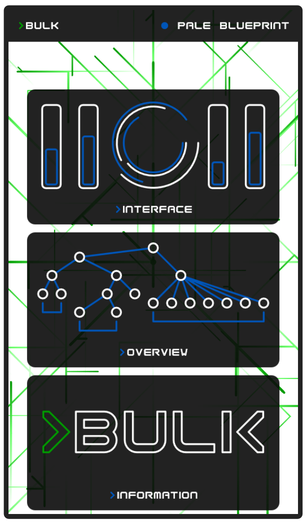
The society initial situation and its operating rules (i.e. routines that keep the Laws object **(3.7.1)**[A.1] as true) are contained in a PHP file. This file (which should be considered as a program entry) is read by the Bulk to populate the database and to indicate which variables should be monitored.

3 Views

3.1 Blocks

The Bulk graphical interface proposes the division of information into multiple blocks of fixed width. This approach meets the expectations of the responsive web design, making pages render well on a variety of devices.





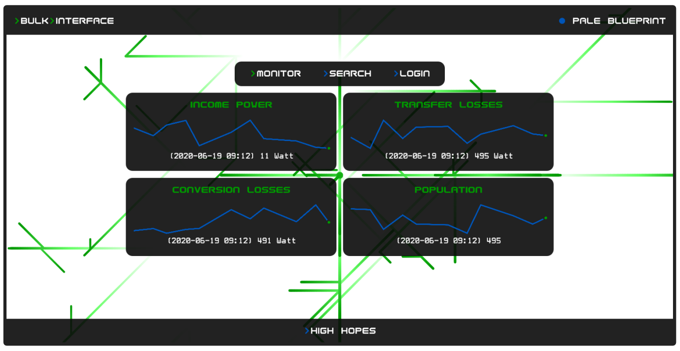
3.2 Bulky

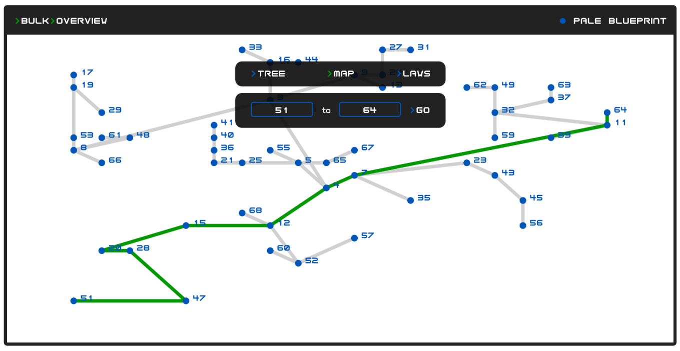
Bulky was designed as an assistant for Bulk visitors and users. He will ultimately be the representation of the AI associated with the Bulk.

```
 !   !
(> <)
 /|\
_/\_
```

3.3 Figures and maps

Figures and maps are generated on the client side by JavaScript. It is planned to study the possibility of executing on the client-side the optimization algorithms to avoid overloading the server and to display changes to the database in real time without having to refresh the page.





4 Controllers

4.1 Optimization

The first and most important optimization algorithm implemented is that of Dijkstra [A.4.1].

RepresentativeManager.php

```php
abstract class Optimization {

  public static function kosaraju($system, $energy) { }

  public static function dijkstra($sender, $input, $receiver, $output) {
    if ($sender==$receiver && $input==$output) return array();
    $channel_manager = new \models\ChannelManager(\models\Settings::db());

    function cost(\models\Channel $channel) {
      return 1/$channel->getEfficiency();
    }

    $nodes = array();
    $channels = $channel_manager->getList(array("sender"=>$sender,
                                                "input"=>$input));
    if (sizeOf($channels)==0) { return array(); }
    foreach ($channels as $channel) {
      array_push($nodes, array($channel->getReceiver(), // system
                               $channel->getOutput(), // energy
                               cost($channel), // cost
                               $channel, // from
                               FALSE)); // explored
    }
    $stop = FALSE;
    while (!$stop) {
      // Search for the unexplored vertex of minimal cost:
      $min_cost = INF;
      foreach ($nodes as &$n) {
        if (!$n[4] && $n[2]<$min_cost) {
          $min_cost_node = &$n;
          $min_cost = $n[2];
        }
      }

      $min_cost_node[4] = TRUE; // Mark the vertex as explored.
      $channels = $channel_manager->getList(array("sender"=>$min_cost_node[0],
                                                  "input"=>$min_cost_node[1]));
      foreach ($channels as $channel) {
        $is_in_nodes = FALSE;
        foreach ($nodes as &$n) {
          if ($n[0]==$channel->getReceiver() && $n[1]==$channel->getOutput()) {
            $is_in_nodes = TRUE;
            $new_cost = $min_cost_node[2]+cost($channel);
            if ($n[2]>$new_cost) {
              $n[2] = $new_cost;
              $n[3] = $channel;
              $n[5] = &$min_cost_node;
            }
          }
        }
        if (!$is_in_nodes) {
          array_push($nodes, array($channel->getReceiver(),
                                   $channel->getOutput(),
                                   $min_cost_node[2]+cost($channel),
                                   $channel,
                                   FALSE,
                                   &$min_cost_node));
        }
      }
      $stop = TRUE;
      foreach ($nodes as $node) {
        if (!$node[4]) { $stop = FALSE; }
      }
      if ($min_cost_node[0]==$receiver && $min_cost_node[1]==$output) {
        $stop = TRUE;
      }
    }
    // Recovery of the path.
    if ($min_cost_node[0]==$receiver && $min_cost_node[1]==$output) {
      $channels = array();
      $step_node = $min_cost_node;
      array_push($channels, $step_node[3]);
      while (sizeof($step_node)==6) {
        $step_node = $step_node[5];
        array_push($channels, $step_node[3]);
      }
      return $channels;
    } else {
      return array();
    }
  }
}
```

4.2 Cron task

A cron task is programmed to load a specific page of the Bulk on a daily basis. This is how, for the moment, the monitor performs regular backups.

4.3 Settings

From the administrator interface it is possible to reset the database (clear the tables) and repopulate it by calling a function defined in the file specific to the chosen society.

A References

A.1 Entropy Economy System

🌐 `https://paleblueprint.org/views/projects/ees/report/2020-04-20.pdf`

A.2 Languages

A.2.1 PHP

🌐 `https://www.php.net`

A.2.2 MySQL

🌐 `https://www.mysql.com`

A.2.3 World Wide Web Consortium

🌐 `https://www.w3.org`

A.3 Design patterns

A.3.1 Model–view–controller

🌐 `https://en.wikipedia.org/wiki/Model-view-controller`

A.3.2 Post-redirect-get

🌐 `https://en.wikipedia.org/wiki/Post/Redirect/Get`

A.4 Algorithms

A.4.1 Dijkstra

🌐 `https://en.wikipedia.org/wiki/Dijkstra’s_algorithm`