Kampach*, c'est, en langue maya yucatèque, la douleur aux lombaires, le lumbago. On lit souvent que le coût énergétique de l'architecture monumentale maya était moindre que ce que l'on peut imaginer de manière intuitive, et c'est l'une des conclusions de ce travail. Il est souhaitable d'écarter fermement les estimations fantaisistes qui peuvent circuler sur la construction monumentale, mais il ne faut pas, pour autant, oublier que les pierres, le sascab, la chaux, les linteaux, l'argile, étaient transportés à dos d'homme. La construction monumentale des Mayas ne se limite pas à des lignes de calculs. Ce fut aussi de la sueur, de la poussière et du sang.*

Voici le mode d'emploi de l'application. Rappelons qu'elle a été pensée pour fluidifier le calcul de coût de main-d'œuvre dans le contexte de l'architecture du Classique ancien dans les Basses Terres centrales. C'est pour ce genre de construction que l'automatisation des calculs est la plus adaptée et performante. Mais cet outil a aussi vocation à être utilisé pour d'autres sites antiques, y compris hors du contexte maya.

1 Principe

Kampach définit un ensemble d'objets qui permettent de modéliser un site archéologique, afin de calculer le coût de main d'œuvre nécessaire à sa construction. Il y a deux catégories d'objets : ceux qui possèdent un coût (les bâtiments et les activités), et ceux, plus abstraits, qui permettent de faire le lien entre ces derniers (les formes géométriques et les *input*). Voici plus précisément la liste de ces objets :

- Les bâtiments. Un bâtiment possède une forme, et de manière optionnelle des structures sub. Son coût est celui de la construction du volume de remblai, ainsi que du volume et de la surface de parement. En présence de structures sub, leur volume est retranché au volume de remblai.
- Les activités, divisées en deux catégories : le transport et la construction. Le coût d'une activité de construction dépend linéairement de la quantité de matière à construire. Celui d'une activité de transport dépend de la distance à parcourir, des vitesses à vide et chargé, et de la quantité transportée par voyage.
  Une activité peut avoir d'autres activités en entrée : avant de construire un mur il faut transporter des pierres, et avant cela les tailler.
- Les formes géométriques, associées aux bâtiments. Les formes disponibles sont les parallélépipèdes, les prismes, les pyramides tronquées et les escaliers.
- Les objets de type *input*. Ils explicitent la dépendance entre un bâtiment et les différentes activités nécessaires à sa construction, ainsi qu'entre une activité et les sous-activités qui lui sont nécessaires. Ce modèle permet de définir tout type de dépendance entre activités ; à ce jour, le seul type de dépendance implémenté est la dépendance linéaire.

Pour mieux comprendre le rôle de l'objet *input*, imaginons par exemple la construction des murs d'une pyramide. L'objet pyramide va permettre de calculer le volume des murs à construire, en mètres-cube. D'autre part, on connaît le coût de l'activité de construction, en jours-hommes par kilogramme de pierre. L'objet *input* fait le lien entre les deux, en précisant d'une part que l'activité de construction est destinée au volume de parement de la pyramide, et en spécifiant d'autre part la densité de la pierre, en kilogramme par mètre-cube, nécessaire pour calculer la masse de pierre nécessaire à partir du volume des murs de la pyramide.

Une fonctionalité intéressante de *Kampach* est la manipulation de quantités avec des bornes d'incertitude : lorsqu'on ne connaît pas précisément la valeur d'une quantité, on peut l'écrire sous la forme d'une moyenne assortie d'une valeur minimale et maximale. Ces incertitudes sont alors propagées dans les calculs, ce qui permet de connaître à tout moment les valeurs extrémales des quantités estimées.

2 Installation

Kampach est une application en cours de développement qui nécessite Python 3. Voici la procédure à suivre pour l'installer :

1. Télécharger *Kampach* : `https://github.com/efroustey/kampach` (lien *Clone or download* puis *Download ZIP*).
2. Télécharger *Pint*[1] : `https://github.com/efroustey/pint`.
3. Dézipper les deux fichiers, et déplacer le dossier `pint-master/pint` dans `kampach-master`.
4. Les calculs peuvent être lancés depuis le dossier `kampach-master` grâce au fichier `run.py` (voir ci-dessous).

3 Utilisation

Les données décrivant le site archéologique doivent être écrites dans un fichier au format XML (voir paragraphe suivant), ce qui peut être fait avec tout éditeur de texte (par exemple Notepad++). Ce fichier doit ensuite être placé au même endroit que le fichier `run.py`. Il suffit ensuite de lancer le fichier `run.py`, soit par un double-clic si un programme qui exécute du code Python est configuré pour ce type de fichier, sinon en ouvrant une fenêtre de commande, puis après s'être placé dans le répertoire contenant le fichier, exécuter `python run.py`. Lorsque cela est demandé, il faut alors entrer le nom du fichier XML décrivant le site archéologique, et taper sur Entrée. Le résultat des calculs s'affiche à l'écran.

1. *Pint* est un module de gestion d'unités physiques. *Kampach* en utilise une version modifiée. La version originale se trouve ici : `https://github.com/hgrecco/pint`

3.1 Écriture des quantités

Les unités physiques sont gérées par le module *Pint*, et peuvent être écrites sous la forme `"1 meter"`. Les unités disponibles se trouvent dans le fichier `pint/default_en.txt`. Les unités peuvent être multipliées, divisées et élevées à une puissance : `"1 kilogram / meter ** 2"` signifie un kilogramme par mètre-carré.

Pour spécifier des valeurs minimale et maximale, il faut les ajouter entre crochets, après une virgule, et séparées par un point-virgule : `"1 meter, [0.9 ; 1.2]"`.

3.2 Le format XML

Un fichier XML est un fichier de texte qui repecte certaines conventions pour décrire des objets. On appelle *balise* ce qui est écrit entre crochets ; par exemple la première balise à écrire pour commencer à décrire un site est `<Site>`.

Les balises s'ouvrent et se ferment : après `<Site>`, on écrit tout ce qui est nécessaire pour décrire le site. À la fin, on referme la balise avec `</Site>`. Si une balise est fermée juste après avoir été ouverte, on peut la simplifier : `<Cuboid></Cuboid>` peut être écrit `<Cuboid />`.

Par convention, la première ligne d'un fichier XML doit contenir le texte suivant : `<?xml version="1.0" ?>`. La seconde ligne doit contenir une balise (du type que l'on souhaite), qui se ferme à la dernière ligne du fichier.

On peut ajouter des *attributs* aux balises : à l'intérieur des crochets, après le nom de la balise, un attribut s'écrit sous la forme *nom="valeur"*. L'ordre des attributs n'a pas d'importance. Par exemple, pour donner un nom au site sur lequel on travaille, on écrit `<Site name="Naachtun">`.

3.3 Balises disponibles

Kampach met à disposition un certain nombre de balises, qui correspondent aux objets que l'on modélise.

3.3.1 Objets ayant un coût

Les balises des objets ayant un coût sont celles qui représentent les bâtiments : `<Site>`, `<SuperBuilding>`, `<Building>` ; et les activités : `<ProductionActivity>`, `<TransportActivity>`. Toutes ces balises doivent avoir un attribut `name` qui décrit leur nom (nécessaire lors de l'affichage des résultats de calcul). A l'intérieur d'une telle balise (c'est-à-dire entre la balise d'ouverture et celle de fermeture), on peut placer une balise `<Inputs>`, qui contient une liste de balises de type *Input* (voir plus bas) décrivant les bâtiments ou activités nécessaires à la construction du bâtiment/réalisation de l'activité décrite.

Voici en détail les balises disponibles :

— `<Site>` décrit un site archéologique. Cette balise n'a besoin d'aucun attribut hormis `name`. C'est en général la première balise que l'on écrit dans

un fichier. À l'intérieur, sous sa balise `<Inputs>`, on met des `<Building>` ou `<SuperBuilding>`.

— `<SuperBuilding>` ou "super bâtiment" représente un bâtiment composé d'autres bâtiments. Cela sert à visualiser le coût d'un ensemble de bâtiments. Comme `<Site>`, cette balise n'attend pas d'attribut hormis `name`.

— `<Building>` représente un bâtiment, qui possède une forme géométrique et éventuellement des structures sub dont le volume est retranché au volume de remblai. Pour définir la forme géométrique, on place une balise `<Shape>` à l'intérieur de `<Building>`, et une balise de type géométrique à l'intérieur de `<Shape>`. De même, les structures sub sont définies dans une balise `<Substructures>`, où l'on place une ou plusieurs balises de type géométrique.

— `<ProductionActivity>` représente une activité dont le coût évolue linéairement en fonction d'une certaine quantité de matière à traiter : pour une quantité Q, le coût C est de la forme $C = aQ + b$. Typiquement, il s'agit des activités de construction ou de transformation de matière. Deux attributs doivent être précisés en plus de `name` : `marginal_cost`, égal au coefficient a, et `fixed_cost`, égal à b. S'il ne sont pas renseignés, par défaut `marginal_cost` vaut 1 et `fixed_cost` vaut 0.

— `<TransportActivity>` représente une activité de transport de matière. Les attributs à spécifier sont la distance à parcourir `distance`, la quantité transportée par voyage `amount_per_travel`, et les vitesses à vide et chargé `speed_empty` et `speed_loaded`.

3.3.2 Objets géométriques

Les objets géométriques définissent la forme des bâtiments et permettent de calculer les volumes et surfaces associés. Les côtes à rentrer en attributs sont spécifiques aux types de géométries, à l'exception de l'épaisseur de parement (attribut `finish_thickness`), qui est une propriété commune à toutes les formes. Les balises disponibles pour décrire les géométries sont :

— `<Cuboid>`, qui représente un parallélépipède. Hormis `finish_thickness`, les attributs à spécifier sont la longueur `length`, la largeur `width`, et la hauteur `height`.

— `<Prism>`, qui représente un prisme (par exemple une toiture). Les attributs sont la largeur `width`, la profondeur `depth` et la hauteur `height`.

— `<TruncatedPyramid>`, qui représente une pyramide tronquée. Les attributs sont les longueurs et largeurs à la base et au sommet `bottom_length`, `bottom_width`, `top_length`, `top_width`, ainsi que la hauteur `height`.

— `<Stairs>`, qui représente des escaliers. Les attributs à renseigner sont la largeur de la première marche `bottom_length`, la largeur de la marche palière `top_length`, la profondeur de la surface en contact avec le sol `bottom_width`, la profondeur de la marche palière `top_width`, la hauteur `height`, et l'emprise au sol `depth`.

3.3.3 Objets *Input*

Les balises de type *Input* sont placées à l'intérieur d'une balise `<Inputs>` d'un objet ayant un coût (l'objet parent, un bâtiment ou une activité), et contiennent elle-même une balise d'une activité (l'activité-enfant). Elles servent à permettre, dans les calculs, de déterminer la quantité de matière devant être produite par l'activité enfant afin de satisfaire les besoins de l'activité ou du bâtiment parent. La seule balise disponible est `<LinearInput>`, elle décrit une dépendance linéaire entre le parent et l'enfant. Ses attributs sont :

- `target_amount` : nécessaire lorsque l'objet parent est un bâtiment, cet attribut spécifie le volume ou la surface cible de l'activité enfant. Les valeurs possibles sont :
  - `total_volume` : le volume total,
  - `fill_volume` : le volume de remblai,
  - `finish_volume` : le volume de parement,
  - `total_finish_area` : l'aire extérieure totale,
  - `top_finish_area` : l'aire extérieure horizontale,
  - `walls_finish_area` : l'aire extérieure verticale ou inclinée.

  On peut aussi entrer directement une valeur numérique, par exemple `1000 meter ** 3` ou `1000 meter ** 2`.
- `marginal_amount` : la quantité de matière que doit produire l'activité enfant pour une unité de quantité attendue par l'activité ou le bâtiment parent. Il s'agit typiquement d'une densité (volumique ou surfacique) de matière lorsque le parent est un volume ou une surface d'un bâtiment, et que le coût de l'activité-enfant de construction associée se mesure suivant la masse à produire.

  S'il n'est pas spécifié, `marginal_amount` vaut par défaut 1. Tel est le cas entre deux activités dont le coût se mesure dans la même unité, et qu'il n'y a pas de perte au passage de l'une à l'autre.
- `fixed_amount` : une quantité fixe, indépendante du volume de matière, associée au passage d'une activité à l'autre (positive pour indiquer une éventuelle perte). En général, cette valeur n'est pas spécifiée et vaut zéro.

3.4 Example

Voici un exemple de fichier XML où l'on modélise un site contenant un seul bâtiment dont la forme est une pyramide tronquée, avec une structure sub de la forme d'un parallélépipède surmmonté d'un prisme. On définit les activités nécessaires au remplissage du volume de remblai (transport puis tassement de terre), à la construction des murs et à leur enduit.

```
<?xml version="1.0" ?>
<Site name="A first archeological site">
 <Inputs>
  <Building name="A first building">
   <Inputs>
    <LinearInput marginal_amount="1000 kilogram / meter ** 3"
```

```
    target_amount="fill_volume">
     <ProductionActivity marginal_cost="2000.0 work_day / kilogram,
     [2000.0 ; 2000.0]" name="Earth packing">
      <Inputs>
       <LinearInput>
        <TransportActivity amount_per_travel="50 kilogram, [50 ; 50]"
        distance="100 meter, [100 ; 100]" name="Earth transporting"
        speed_empty="5 kph, [5 ; 5]" speed_loaded="2 kph, [2 ; 2]"/>
       </LinearInput>
      </Inputs>
     </ProductionActivity>
    </LinearInput>
    <LinearInput marginal_amount="2000.0 kilogram / meter ** 3, [2000.0 ;
    2000.0]" target_amount="finish_volume">
     <ProductionActivity marginal_cost="1000.0 work_day / kilogram,
     [1000.0 ; 1000.0]" name="Wall building"/>
    </LinearInput>
    <LinearInput marginal_amount="10.0 liter / meter ** 2, [10.0 ; 10.0]"
    target_amount="total_finish_area">
     <ProductionActivity marginal_cost="0.1 work_day / liter, [0.1 ;
     0.1]" name="Plaster laying"/>
    </LinearInput>
   </Inputs>
   <Shape>
    <TruncatedPyramid finish_thickness="0.5 meter, [0.5 ; 0.5]"
    bottom_length="30 meter, [30 ; 30]" bottom_width="20 meter, [20 ;
    20]" height="10 meter, [10 ; 10]" top_length="10 meter, [10 ; 10]"
    top_width="5 meter, [5 ; 5]"/>
   </Shape>
   <Substructures>
    <Cuboid height="6 meter, [5 ; 5]" length="5 meter, [5 ; 5]" width="5
    meter, [5 ; 5]"/>
    <Prism width="5.000 meter, [5.000 ; 5.000]" depth="5.000 meter, [5.000
    ; 5.000]" height="5.000 meter, [5.000 ; 5.000]"/>
   </Substructures>
  </Building>
 </Inputs>
</Site>
```