Utiliser et transformer des critères en esquisses de plan

Nathan Beyler
Séminaire Activités et instrumentation de la conception
Ecole nationale supérieure d'architecture de Paris la Villette
N° 15679 - 08/01/2019

Sommaire

Contexte et enjeux

1) Utiliser et transformer des critères en esquisses de plan

Ce mémoire a pour but d'établir une relation entre critères ou paramètres et esquisses de plan. Cette thématique n'est pas nouvelle, on retrouve des travaux en ce sens depuis le milieu des années 1960 avec la démocratisation de l'informatique.

L'utilisation d'un ordinateur en architecture n'est plus à remettre en question, de nos jours la plupart des architectes utilisent quotidiennement des logiciels de CAO , mais jusqu'à quel point la machine peut nous aider dans le processus de conception ? On voit de plus en plus la popularisation de logiciels d'architecture paramétriques comme GrassHopper ou Dynamo, utilisant donc des paramètres ou plus généralement une formalisation mathématique ou informatique afin de produire diverses architectures. On peut cependant observer que l'utilisation la plus courante de ces logiciels concerne une optimisation des formes et ne touche finalement que très peu la phase de conception d'un projet. On retrouve quelques traces d'optimisation de plan à travers certains plug-ins comme Galapagos pour GrassHopper, mais cela concerne plus l'optimisation justement que la génération.

L'idée de ce mémoire est donc de chercher à créer un logiciel de génération d'esquisses de plan à l'aide de critères via une interface graphique. Pour ce faire, il est dans un premier temps nécessaire de présenter les bases du programme, ses constituants, son fonctionnement, puis de définir les critères qui interviendront dans la génération, leur modélisation informatique ainsi que la formalisation de notions architecturales et enfin d'expérimenter le programme sur quelques exemples concrets.
Il serait tentant de parler, à propos de ce mémoire, d'architecture paramétrique étant donné qu'il s'agit de concevoir les prémisses de projets architecturaux à l'aide de paramètres, mais cette notion désigne plus la création de formes complexes en se basant sur l'utilisation de données. Les précurseurs dans le domaine ont préféré utiliser d'autres termes comme allocation spatiale, approche générative ou encore le concept anglais de Planning que l'on pourrait traduire un peu approximativement par planification ou aménagement.

Ces quelques paragraphes liminaires laissent à penser que le programme est finalisé, ce n'est malheureusement pas le cas et certains aspects qui seront présentés dans la suite du document ne seront que théoriques, ces notions sont néanmoins importantes et ont, d'une façon ou d'une autre, participé à modeler l'ensemble actuel.
Il est important également de garder à l'esprit que le but n'est pas de demander à un ordinateur de produire des plans d'architecte, mais d'automatiser la recherche d'orientation de projet à l'aide de critères définis par l'utilisateur et d'un processus orienté, lui aussi, par l'utilisateur, il est d'ailleurs inconcevable actuellement d'imaginer une démarche architecturale un tant soit peu complexe sans architecte, le cerveau humain reste inégalé dans bien des aspects de la conception.

2) État de l'art historique et actuel

On pourrait remonter les prémisses de l'architecture paramétrique aux travaux de Gaudi autour de ses observations de la nature et notamment ses maquettes filaires de la Sagrada Familia, le biomimétisme étant souvent associé à des paramètres discrets. Un des projets précurseurs dans le domaine est le projet SEEK du MIT supervisé par Nicholas Negroponte et Leon B. Groisser en 1970.
Dans SEEK, des gerbilles sont placées dans un espace clos représentant une ville. La ville est modélisée à la fois informatiquement, mais également physiquement grâce à des petits cubes. Les gerbilles, en se déplaçant, mangeant, etc. modifient la ville, bougeant les cubes, faisant tomber des tours, mais celle-ci est automatiquement reformée par des bras articulés ayant pour modèle la maquette informatique.

Pour retrouver les prémisses de l'architecture paramétrique ou encore de l'optimisation de plans, on peut cependant remonter un peu plus loin dans le temps, vers le milieu des années 60 avec notamment les travaux de Christopher Alexander. Alexander publie en 1964 «De la synthèse de la forme» puis développe les programmes HIDECS (1,2 et 3) associés à sa publication «Four computer programs for the hierarchical decomposition of systems which have an associated linear graph». De par ses recherches, Alexander est l'un des premiers à soulever des problématiques dans le domaine de la conception associées à une formalisation mathématique liée à l'informatique. Cependant, certains considèrent que suite à une formalisation trop abstraite, il réduit l'importance de la problématique architecturale. Il faut cependant garder à l'esprit que l'informatique de l'époque n'en était qu'à ses balbutiements et que la programmation dans les années 60-70 n'avait rien à voir avec l'actuelle.

Peu après Alexander, B. Whitehead s'est intéressé aux traitements de données grâce à des graphes, les arrêtes représentant les coûts de déplacement notamment à travers ses travaux avec O.M Agraa et E. Hafez «Automation of data preparation in computer program for the planing of single storey layout» de 1965 puis 1967. Whitehead rencontre cependant une problématique de taille, les graphes obtenus ne sont pas planaires et donc difficile à utiliser avec les plans de l'époque.
Parallèlement, P.H Levin travaille sur «Use of graphs to decide the optimum layout of buildings» et en juxtaposant les travaux de Levin et Whitehead on peut alors suggérer de résoudre la problématique précédente en aplatissant les graphes par la suppression des arêtes les moins importantes en leur attribuant un coefficient.
Whitehead écrira également avec O.M Agraa «The Planning of Single-Storey Layouts», travaux portant sur l'utilisation de données quantitatives dans des matrices et le traitement de celles-ci. À l'époque, les recherches sur l'optimisation de plans n'utilisent que des données quantitatives et dans ces approches elles sont souvent estimées et ne représentent pas forcément une réalité architecturale, par exemple, Whitehead et Agraa étudient le nombre de pas par salaire respectif des membres d'une catégorie donnée.

Par la suite, les travaux et programmes se sont petit à petit complexifiés. Dans son programme LOKAT et sa publication «A generalized program for transforming relationships values into plans layouts» (1970), Allen Bernholtz utilisera plus d'une dizaine de paramètres. Ces paramètres permettent de pondérer des variables aléatoires et de trouver quelques solutions architecturales acceptables.

On passe alors de «la» solution «aux» solutions, il n'est plus alors question d'optimisation unique ce qui est plus sage si on considère la difficulté de l'Architecture. Les solutions «acceptables» ne sont néanmoins pas suffisantes et à partir de ce moment, les recherches ont commencé à se porter plus sur l'étude de la façon dont les architectes résolvent certaines problématiques plus que sur une formalisation mathématique et informatique.

Ci-dessous quelques systèmes de représentation :

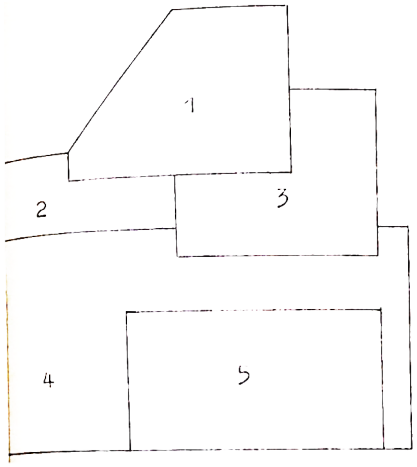
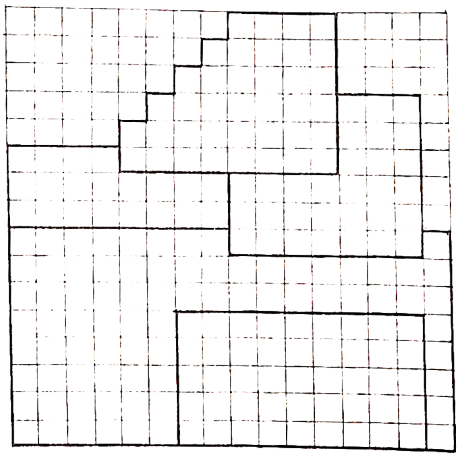
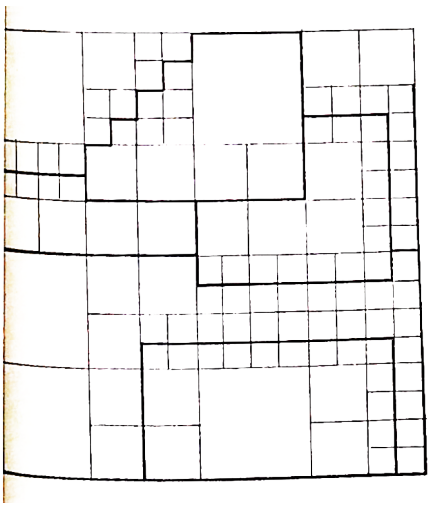


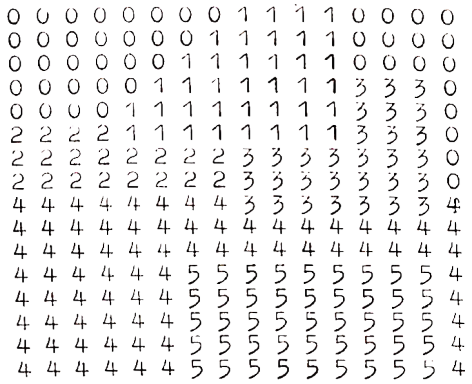
Schéma à représenter



Représentation par grille, représentation la plus utilisée, composée de cellules unitaires.



Représentation par tableaux hiérarchiques. L'espace est divisé puis chaque sous-espace est récursivement divisé en fonction de la finesse exigée.



Représentation de la grille en mémoire par tableau

Schémas issus du Séminaire sur l'allocation spatiale

La création d'interfaces animées incluses dans un système d'exploration «Pavlov» va permettre de faciliter les recherches dans le domaine. Comme expliqué précédemment, les programmes (essentiellement déterministes) sont peu efficaces, J.P Boudier, S. Chalambides, A.M Fourcade et G. Lafue diront même, au cours du séminaire sur l'allocation spatiale d'avril 1973 : « C'est le grand nombre (de solutions) qui interdit la plupart du temps de pouvoir envisager une production exhaustive des solutions, même par les ordinateurs les plus puissants », on peut également souligner la différence entre la programmation de l'époque et les langages actuels qui ont nettement évolué depuis les années 70. Ils développeront alors l'un des premiers systèmes d'intelligence artificielle en architecture en définissant une arborescence utilisant les solutions comme feuilles et les branches correspondant au processus de recherche. L'IA va alors présenter les solutions en faisant des choix d'embranchement par rapport aux critères de l'utilisateur.

Par la suite, la recherche concernant la génération de plan disparait petit à petit au profit d'avancée concernant la modélisation. Cette disparition est essentiellement due à des résultats qui tendent bien souvent vers l'impossibilité de concilier de nombreux paramètres.
En 1993, Greg Lynn, un des pionniers concernant la modélisation, propose des théories architecturales portant sur l'inflexion, le pliage et les courbures. Par la suite, en 1996 il utilisera le terme d'architecture blob (nom emprunté au film Blob de 1956 écrit par Kay Linaker) la forme est alors donnée par l'idée du mouvement et d'informations in situ. Son travail permettra également de développer non seulement des idées de fluidité, d'animation et d'organisation, mais également d'expérimenter la modélisation et la représentation de géométries complexes.
Quelques années plus tard, on observera un tournant majeur pour l'architecture paramétrique et plus largement l'architecture à forme libre grâce au Musée Guggenheim à Bilbao de F. Ghery (1997). Le bâtiment a un impact visuel très fort, utilisant des courbes, une architecture fluide ainsi que des matériaux très représentatifs. Les géométries très complexes utilisées par Ghery se font par le logiciel CATIA, jusqu'à présent cantonné au secteur industriel. Cependant le musée Guggenheim débute d'esquisses et d'une réflexion non informatisées.

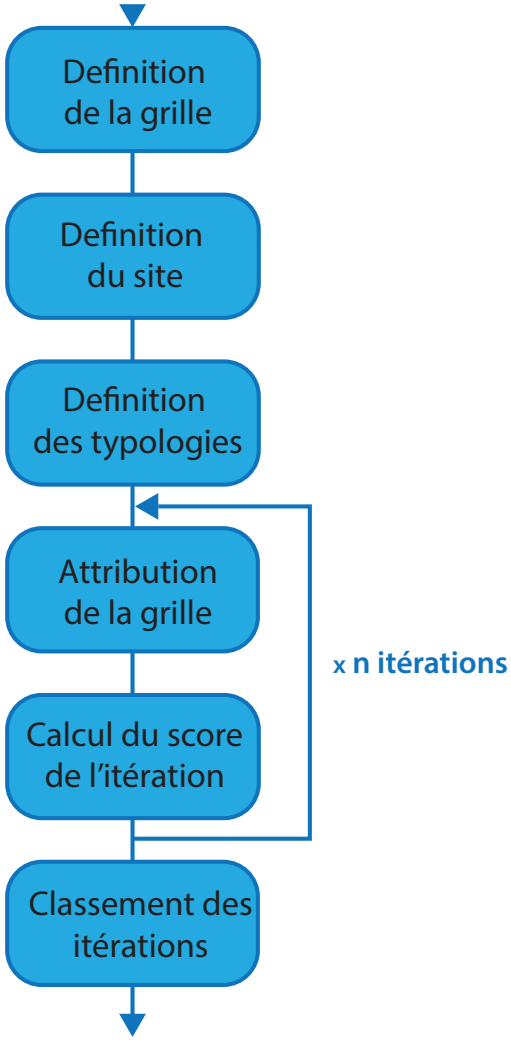
L'architecture paramétrique utilisant alors des paramètres comme base du travail architectural ne se développera que plus tard, notamment concernant la structure, la luminosité ou répondant à des problèmes bien spécifiques. On observe en effet un retour à l'utilisation de logiciels conercnant la conception avec l'apparition de logiciels de modélisation paramétrique comme GrassHopper, Galapagos, Kangaroo (tournant grâce au logiciel Rhinocéros) ou encore Dynamo et Revit. Concernant Galapagos, on retrouve à travers ces logiciels un fonctionnement similaire à la théorie de l'évolution grâce à des itérations de plus en plus précises et donc le début d'une seconde vague de recherches concernant l'optimisation de plans qui mènera certainement vers la génération de plans. Comme exemples actuels de bâtiments utilisant l'architecture paramétrique, on peut citer le fameux Water Cube de Beijing en Chine (PTW Architects), le Beijing National Stadium par Herzog & de Meuron ou encore le Gunagzhou Opera par l'agence Zaha Hadid Architects.

Les bases du programme

Le principe de ce programme est de générer des esquisses de plan optimisées en fonction de paramètres décidés par l'utilisateur. Pour ce faire, les générations sont créées itérativement de façon semi-aléatoire puis comparées entre elles à l'aide d'un score afin de définir la meilleure génération.

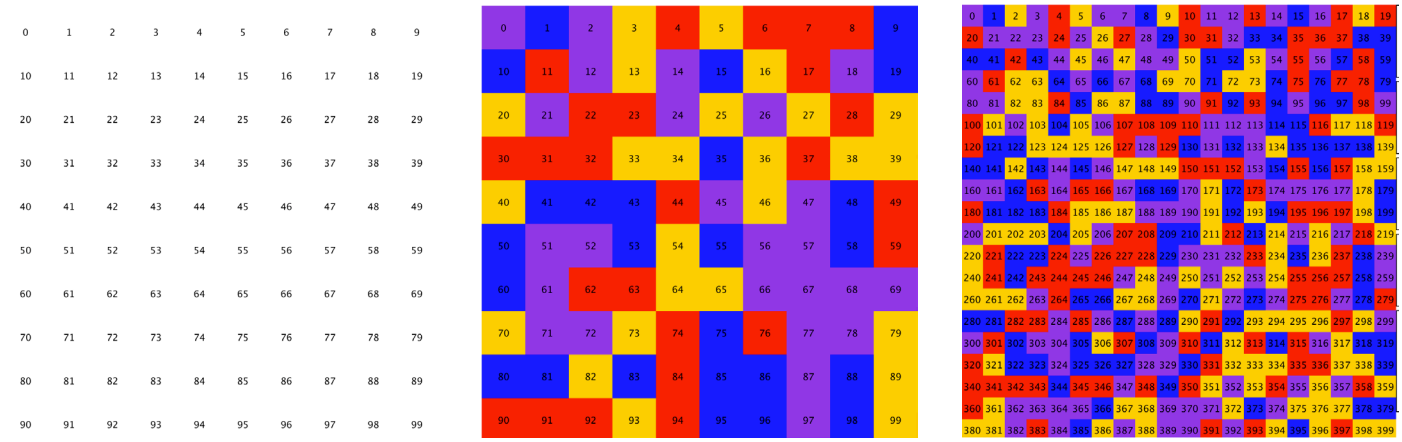
Le programme se base sur une grille de cellules attribuées à des typologies. La grille est un moyen de représentation particulièrement adapté aux plans, la division de celle-ci permettant de jouer sur la précision des esquisses. Les typologies représentent les différents composants du plan, par exemple, pour un logement, on utilisera des typologies comme les chambres, salle de bain, salon, etc. Ces typologies se différencient par de nombreux paramètres de dispositions, de luminosité ou encore de répartition. Les cellules sont les unités de base de la représentation en leur attribuant une typologie.
En intégrant de nombreuses variables aléatoires dans la génération, cela permet d'obtenir des plans différents, prenant des orientations architecturales variées. Plus on génère un grand nombre d'esquisses, plus on aura d'éléments de comparaison et plus le résultat final sera proche de la solution optimale.

Fonctionnement général

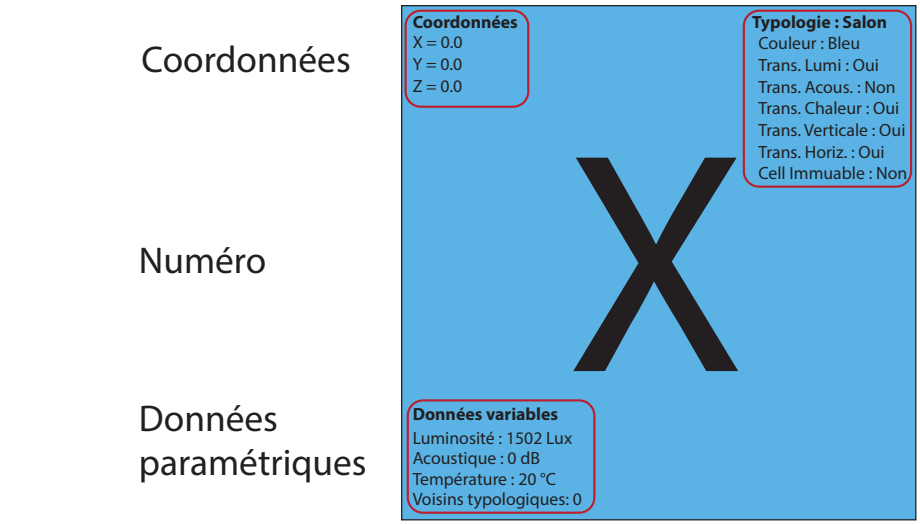


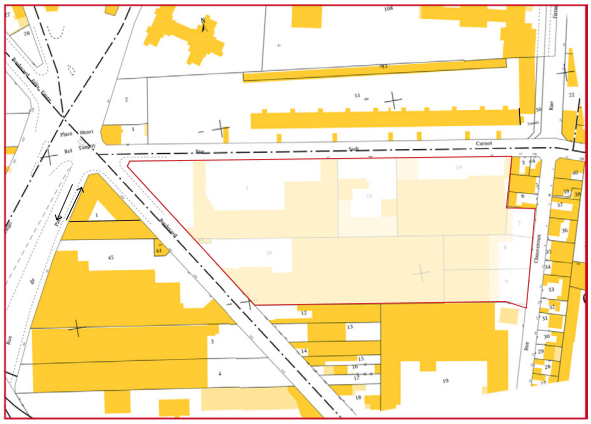
1) Grille et cellules

Le principe de base du programme est l'adaptabilité. En effet, celui-ci doit pouvoir s'adapter aux différents sites . La grille de base comporte 10 x 10 cellules, mais il est possible de définir à la fois le nombre de cellules, mais également la taille de celles-ci et leur répartition, on peut ainsi créer une grille rectangulaire munie de cellules également rectangulaires.

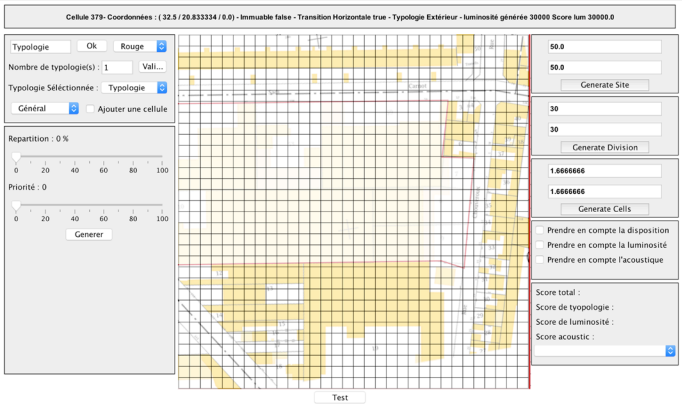


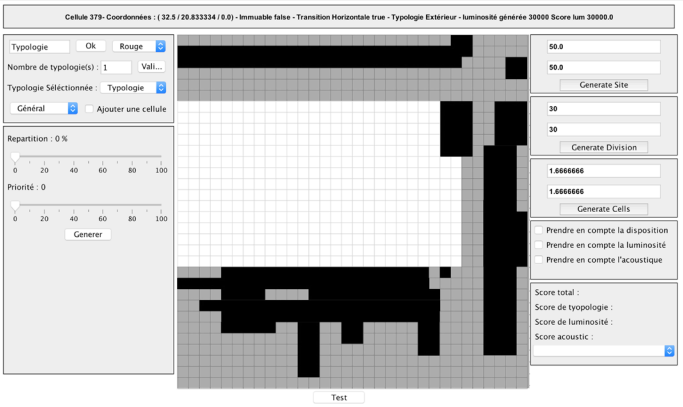
Les cellules possèdent de nombreux paramètres. Certains paramètres sont inhérents à la cellule comme ses coordonnées, son numéro, sa luminosité ou sa typologie, tandis que d'autres proviennent justement de la typologie de la cellule, sa «famille».





Grâce à l'adaptabilité de la grille, il est possible en imposant certaines cellules de représenter le site sur lequel se base le projet. En effet, il est possible d'attribuer certaines typologies comme celles du bâti plein ou du vide extérieur autour desquelles l'attribution sera générée.

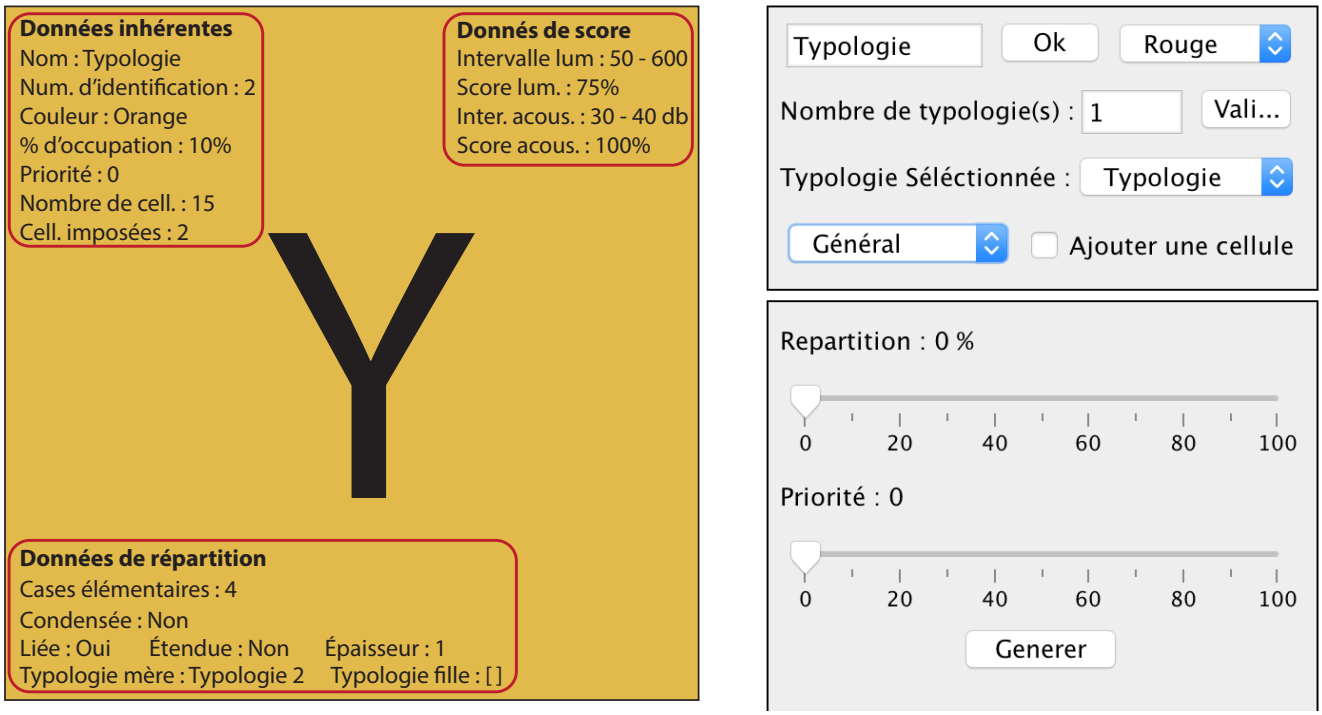




2) Typologies, unités de base du programme

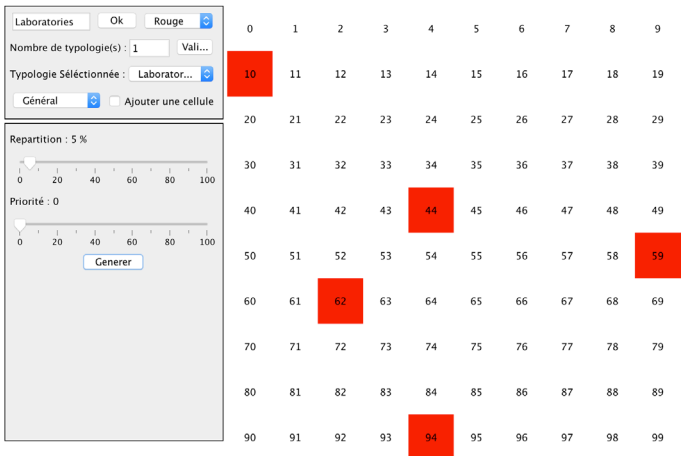
Les typologies sont à la base du programme, mais plus généralement à la base de la conception architecturale. Les intentions d'un projet s'ancrent dans la disposition de celles-ci, leurs relations, leurs dispositions, etc.

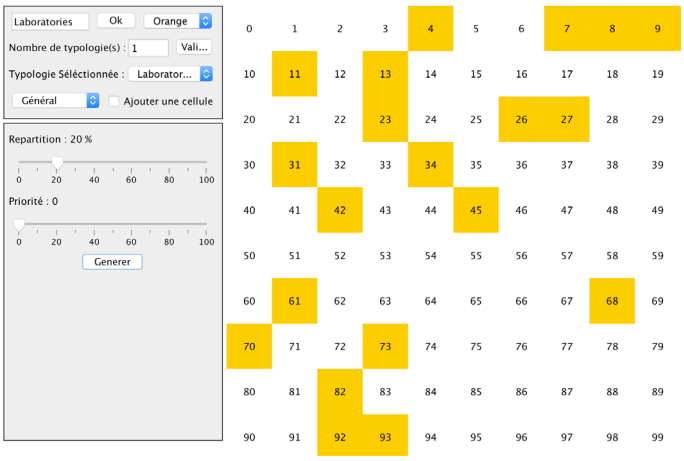
Dans ce programme, les typologies possèdent un nombre important de critères les définissant. Parmi les données inhérentes d'une typologie, on retrouve son nom, son numéro d'identification qui permettra de les manipuler facilement dans le programme, sa couleur de représentation pour les distinguer visuellement, une valeur d'occupation de la grille, une priorité puis des cellules qui sont donc les constituants, les atomes, de la typologie.
La valeur d'occupation de la grille est en pourcentage afin de pouvoir lier différentes analyses préalables à la génération. En effet, il peut être intéressant de faire une étude de répartition de typologie dans de nombreux précédents afin d'avoir une base de départ, un ordre d'idée, pour générer des esquisses de plans. La priorité d'une typologie est utile afin de trancher dans le cas de paramètres contradictoires. Les volontés contradictoires sont à la base de l'abandon de la génération paramétrique dans les années 80, les architectes se sont rendu compte qu'il était parfois impossible de concilier toutes leurs volontés et qu'ils étaient obligés de faire des concessions, ce qui limite l'intérêt de tels programmes. Je reste néanmoins persuadé que les avancées technologiques ainsi que de nouvelles approches de ces questions ravivent l'intérêt de l'informatique en architecture, en est témoin l'architecture paramétrique.



Ci-dessus le diagramme de représentation des typologies et l'interface de paramétrage concernant les typologies

La représentation graphique est la finalité de ce programme, c'est l'esquisse de plan que l'on cherche. Ce sera ensuite la base du travail de l'architecte, c'est le lien direct entre l'informatique et l'architecture.





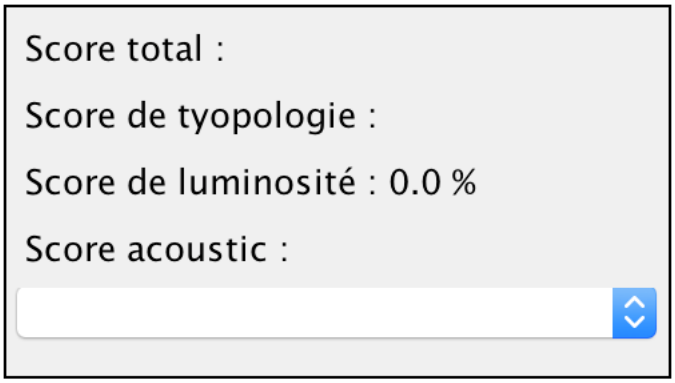
3) Scores et paramètres

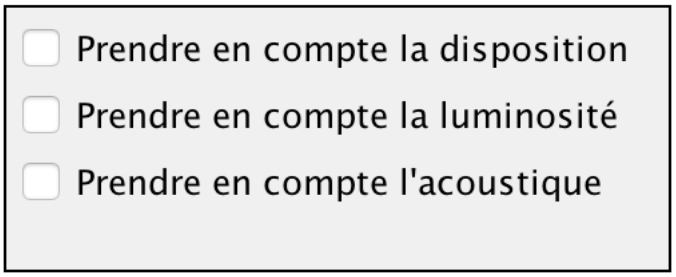
L'objectif du programme étant de déterminer des solutions plausibles et de les hiérarchiser les générations obtenues, il est essentiel de pouvoir comparer les différentes dispositions. Pour rappel, chaque itération possède volontairement de nombreux paramètres aléatoires afin de diversifier les résultats et pouvoir toucher un large spectre de propositions architecturales.

La principale difficulté rencontrée est de savoir comment traduire numériquement des schémas et intentions architecturaux et comment comparer ces données extrêmement diverses. Il est important de discrétiser les résultats en pourcentage, ceci permettra de comparer ensuite les valeurs entre elles malgré la diversification des domaines et des natures de paramètres.

Quels critères comparer ?
Pour l'instant le programme ne permet que de comparer la luminosité et l'agencement des générations.
La luminosité est sans doute le paramètre que l'on retrouve le plus en architecture. Le traitement de la lumière est essentiel dans chaque projet et il est compliqué de ne pas la prendre en compte pour comparer les esquisses. De la même façon, l'agencement, la répartition des différentes typologies correspond au projet en lui même, il est ainsi impossible de ne pas les confronter.

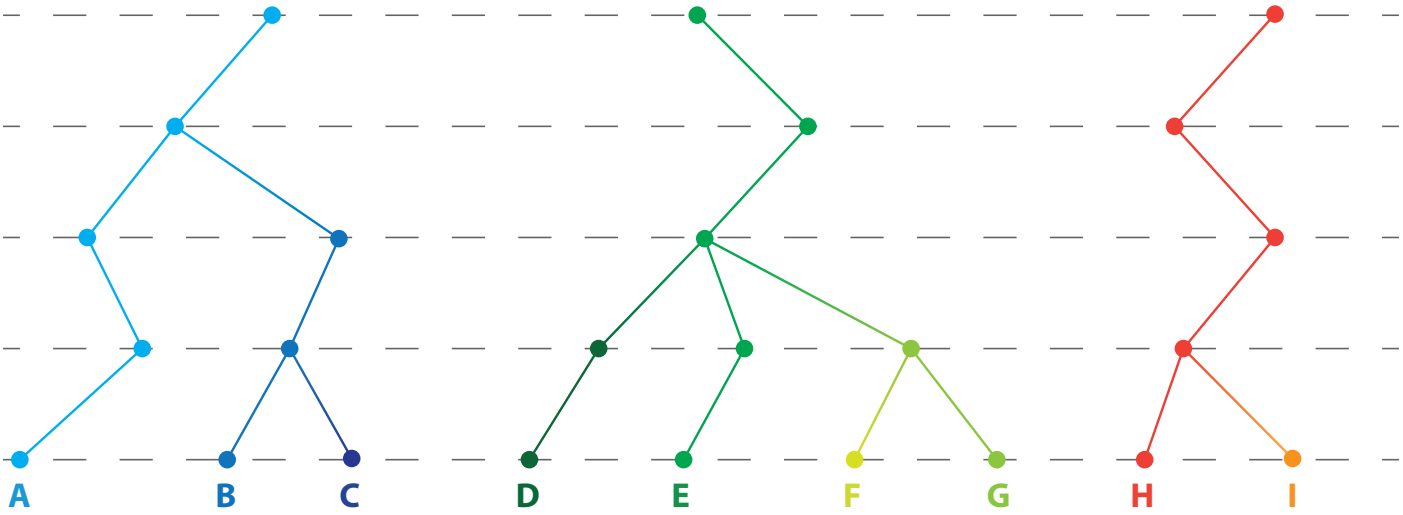
Idéalement, avec plus de temps, il serait intéressant de prendre en compte de nombreux autres critères comme l'acoustique, l'énergétique, les déplacements ou encore le prix des générations. Pour aller plus loin, il faudrait également attribuer des coefficients aux différents paramètres de comparaisons afin de privilégier certaines orientations de projet. C'est également grâce à ces coefficients qu'il sera possible d'affiner les résultats.





Un problème se pose néanmoins : qu'en est-il de la relation entre les plans ? En effet, à fonctionner de cette façon, les esquisses de plan sont générées indépendamment et il n'existe aucune utilisation d'un plan pour en générer un autre, les résultats ne s'affinent pas et rien ne nous dit qu'on qu'avec 10 000 générations on ne passera pas à côté de la 10 001e fondamentalement plus intéressante. Ce qui suit n'a pas été implémenté dans le programme, mais aurait dû l'être à terme. Le but du programme n'est pas de trouver la solution optimale, mais plus de proposer une arborescence de projet, en considérant les esquisses obtenues comme les feuilles et l'algorithme itératif comme le branchage.

L'idée est simple, il faut tout d'abord générer un grand nombre de plans et en calculer les scores respectifs. Chaque score correspond donc à un plan qui correspond lui-même à une suite de choix successifs de l'algorithme. En observant les scores les plus élevés, il est possible de voir des tendances se dégager.
Dans le schéma ci-dessous, chaque point représente un choix de l'algorithme :



L'esquisse A de score S(A) correspond donc à une succession de choix. Mais comment améliorer ces résultats ? L'idée serait de repartir d'un noeud de l'arbre pour explorer diverses possibilités et ainsi favoriser l'obtention d'un meilleur score. Il existe cependant un problème de taille : le calcul de score ne se fait qu'à la fin, il est donc difficile de savoir quel noeud choisir pour obtenir un meilleur résultat. Cela peut sembler évident avec le schéma précédent extrêmement simple, mais étant donné le nombre de choix différents effectués par le programme, beaucoup d'arbres n'auront qu'une seule branche. Pour éviter de traiter chaque arbre, il semble tout d'abord évident d'en choisir un nombre réduit offrant les meilleurs scores.Pour la suite, plusieurs possibilités s'offrent à nous :
- Reprise d'un noeud créant plusieurs branches, cette solution semble cohérente, mais comme expliquée précédemment, de très nombreux arbres n'auront qu'une unique branche

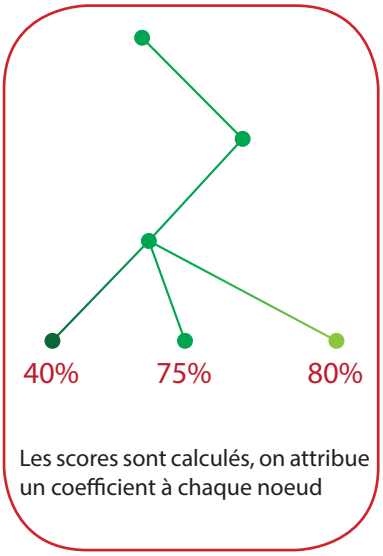
- Création d'un score partiel à chaque noeud, cela nous permettra de savoir quel noeud possède le score le plus élevé et de nous en servir de base pour de nouvelles générations. Ce principe est théoriquement le meilleur, étant donné qu'on choisit le meilleur spécimen de l'arbre pour générer les branches ataviques. Le problème c'est que certains paramètres qui entrent dans le calcul du score nécessitent une génération complète et non partielle pour être pris en compte, et un score sans ces paramètres ne serait pas forcément représentatif.

- Création d'une interface visuelle en remplaçant les noeuds par les esquisses à chaque état correspondant puis en laissant l'utilisateur choisir. Cette solution présente l'inconvénient de ne plus être automatisée, ce qui serait trop long dans le cas d'un très grand nombre de générations différentes, on ne peut demander à l'utilisateur de traiter un millier de plans

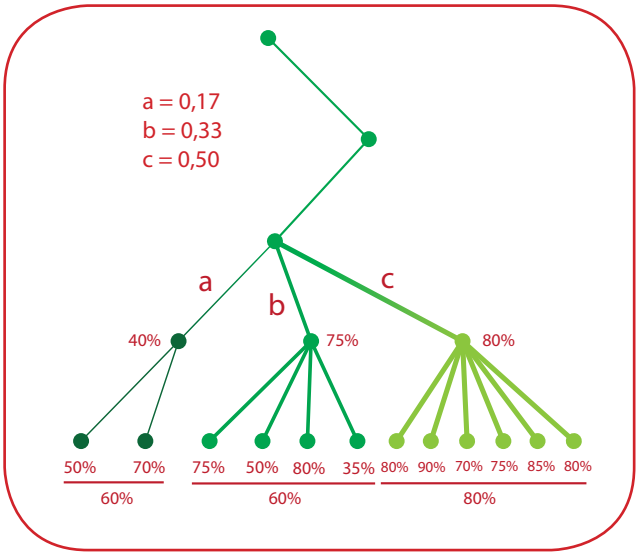
- Création d'un score nodal attribué à chaque noeud en générant une petite quantité de plans à chaque noeud et en attribuant le score moyen de ce petit échantillon. Cette démarche semble

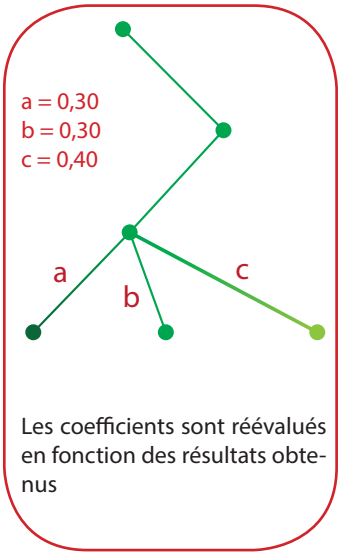
être la plus adaptée, elle reprend le système d'un score partiel qui aurait une cohérence étant donné qu'il n'éliminerait pas de critères. On peut se dire qu'un score moyen n'est pas forcément représentatif et qu'on risque de louper les meilleures esquisses, il a cependant été démontré en statistiques que d'après le calcul de l'intervalle de fluctuation, une étude menée sur 1000 échantillons aurait plus de 95% de chance de présenter un résultat correct à ±3 %, les instituts de sondage contestent cette étude en expliquant que leurs résultats sont encore meilleurs, il n'y a donc pas besoin d'un nombre immense d'échantillons pour avoir un résultat représentatif (on pourrait même se contenter d'un résultat moins précis).

Afin de ne négliger aucune possibilité, il serait également intéressant de coefficienter les branches et de continuer de générer en petite proportion sur les branches non sélectionnées puis de réétalonner les coefficients en fonction des résultats.



Les scores sont calculés, on attribue un coefficient à chaque noeud





Les coefficients sont réévalués en fonction des résultats obtenus

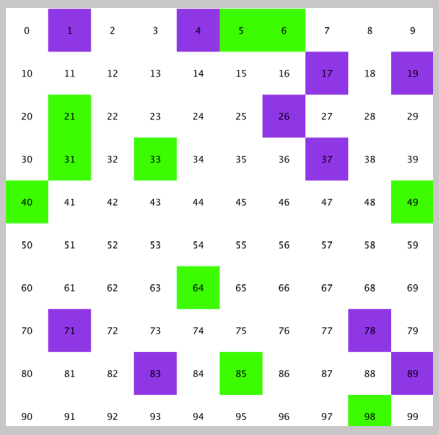
4) Itérations et semi-aléatoire

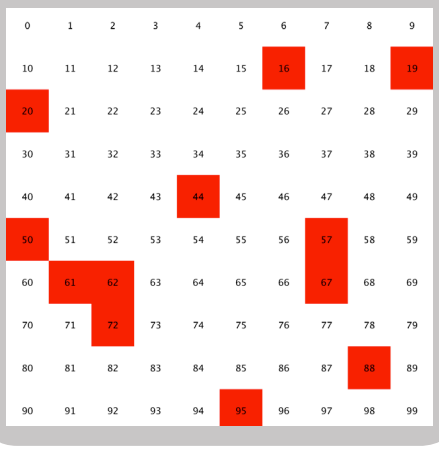
Actuellement, l'algorithme est itératif dans sa façon de procéder, mais pas dans son fonctionnement général, dans le sens où les différentes générations ne sont pas liées les unes aux autres, mais l'attribution des cellules, le calcul des différents scores, etc. sont faits itérativement. L'une des directions du programme serait de pouvoir lier les générations afin de créer une arborescence et distinguer clairement les différentes orientations du projet.

Comme expliqué auparavant, de nombreux éléments aléatoires interviennent dans ces itérations, essentiellement dans l'attribution des cellules (le tracé des esquisses) pour pouvoir diversifier au maximum les résultats et ne pas obtenir deux fois la même chose avec les mêmes paramètres. Si les générations étaient liées, chaque choix aléatoire créerait une nouvelle branche dans l'arborescence.

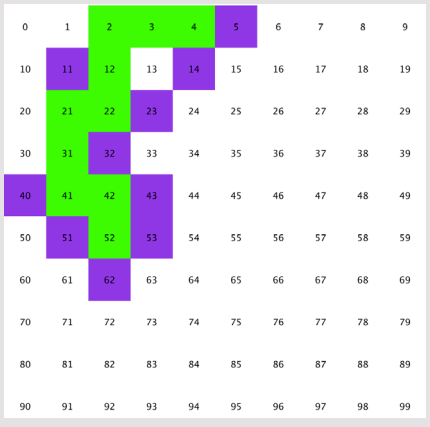
Il est cependant extrêmement peu productif de procéder à une génération complètement aléatoire. Il serait possible de choisir d'attribuer complètement au hasard les cellules, mais moins de 3% des résultats sont un minimum probants (chiffre obtenu sur des essais personnels). Il y a ainsi beaucoup trop de perte de calcul/temps/énergie dans une solution purement aléatoire et il est important de procéder à une itération semi-aléatoire. Pour ce faire, l'attribution des cellules est orientée en fonction des paramètres choisis, mais tout sera détaillé par la suite.

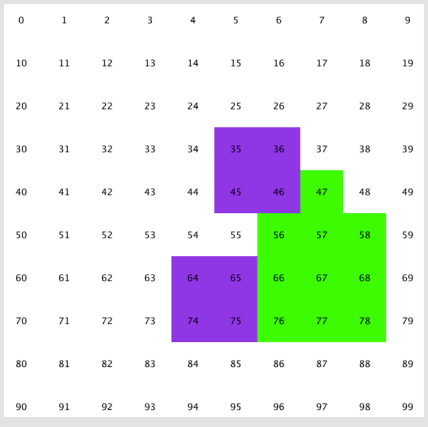
Aléatoire

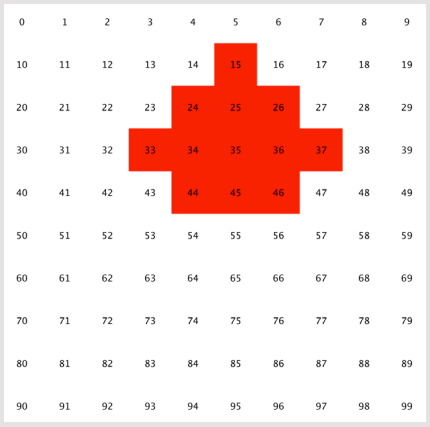


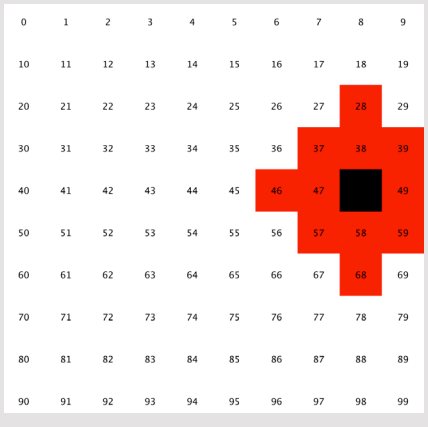


Semi-aléatoire





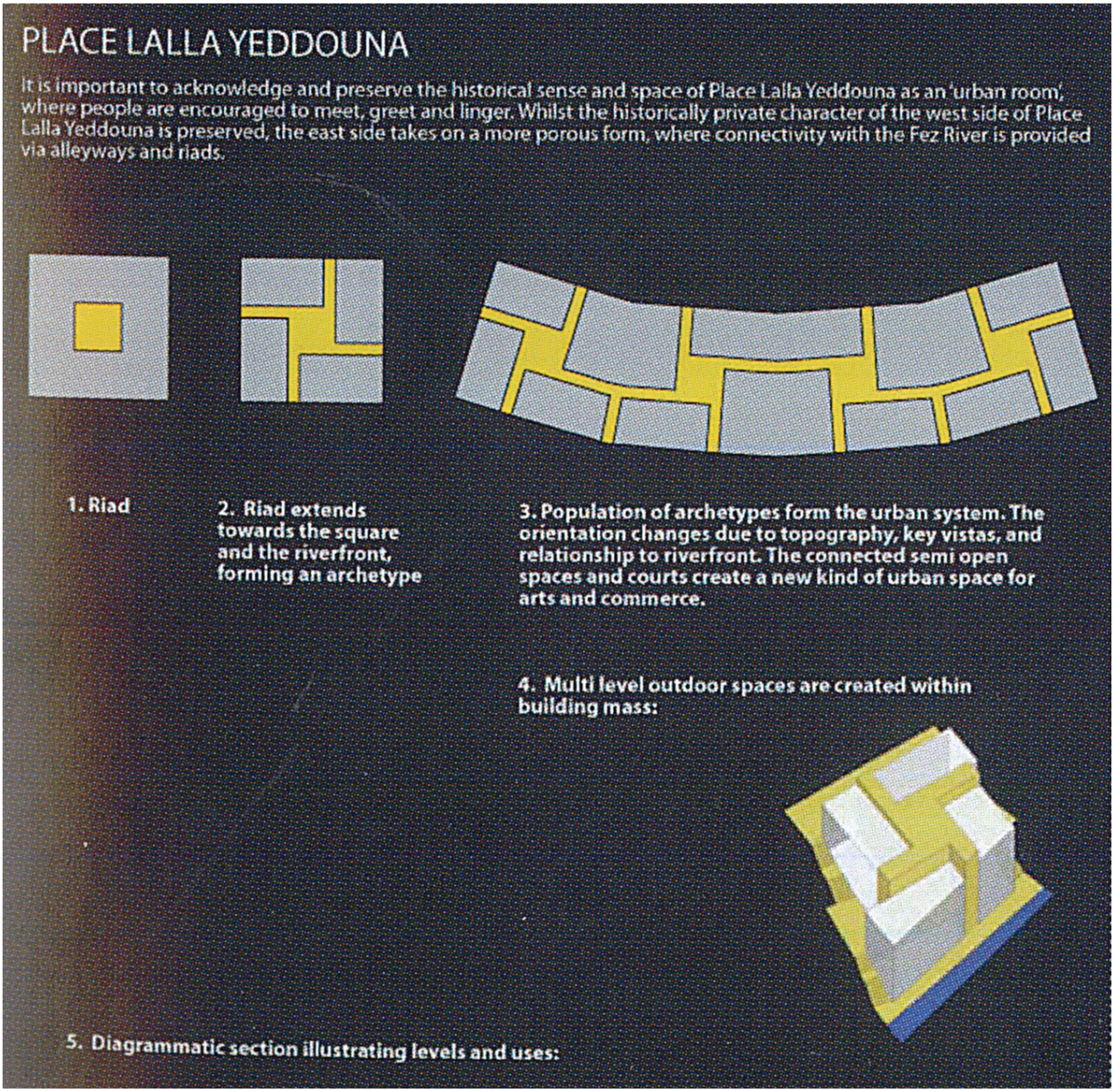




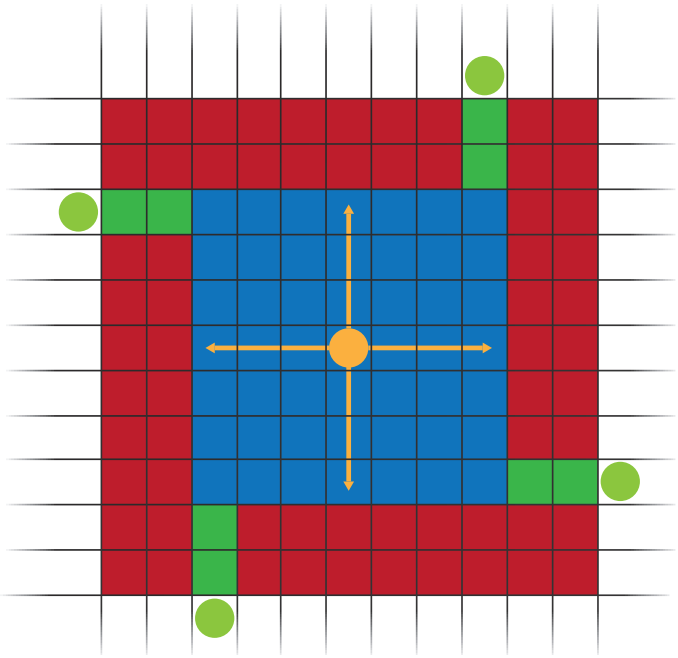
Études de cas

Afin de déterminer quels paramètres prendre en compte pour la comparaison des générations, mais également quelles dispositions prévoir dans l'attribution semi-aléatoire, il m'a paru plus censé d'analyser non pas des plans terminés, mais plus des planches de concours, des diagrammes ou encore un projet personnel dont je connais le développement.

1) Projet Place Lalla Yeddouna, Mossessian & partners architecture

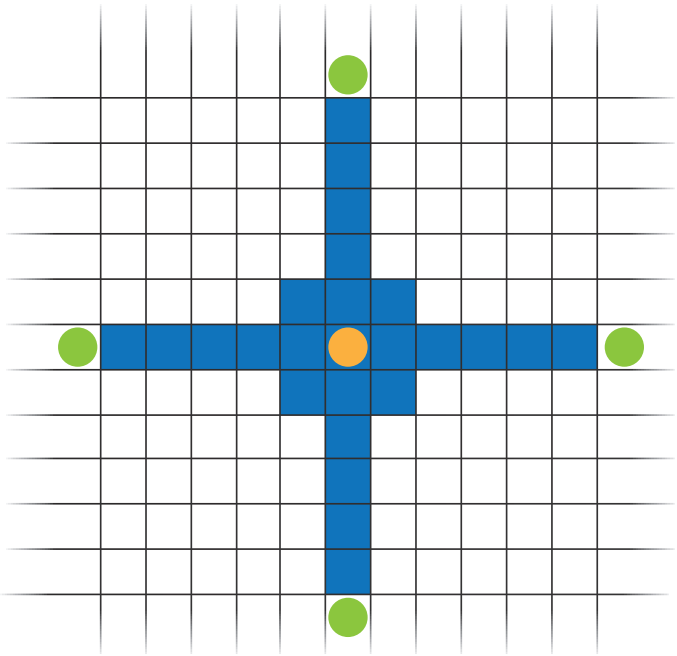


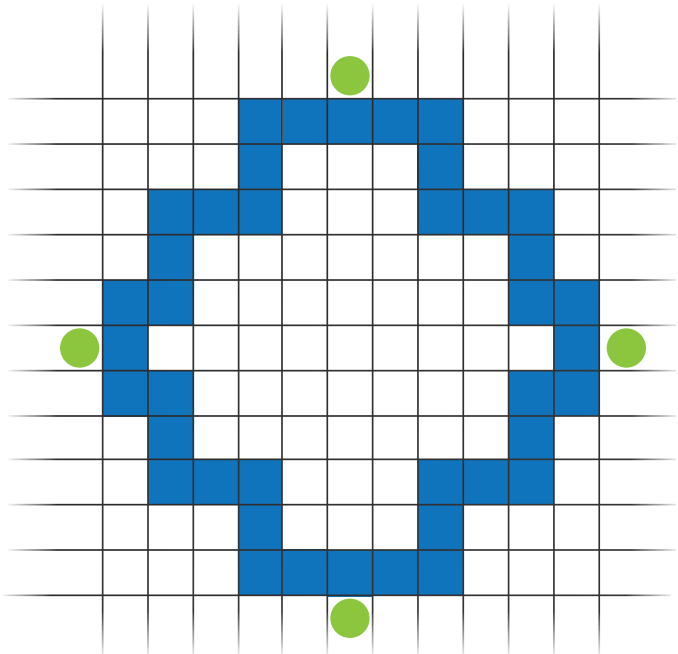
Que peut-on retenir de ce projet ?
Pour leur conception, les architectes ont d'abord considéré la forme typique du Riad marocain et de son patio central, ils ont ensuite lié ce patio aux différentes sorties du Riad et ont répété ce motif plusieurs fois.



Il serait possible de proposer 3 typologies distinctes: le bâti, le vide central et les accès à ce vide. Pour ce faire, on peut définir le bâti comme typologie repoussée par le centre de la grille, le vide central comme au contraire, condensé au centre et les accès comme typologie reliant des points définis (sorties) à la typologie centrale.

Il peut néanmoins être également intéressant de proposer une typologie permettant de relier différents points entre eux (ici, les 4 sorties du modèle de base) puis de permettre au surplus soit d'être rajouté autour du centre de gravité ou de façon plus ou moins aléatoire. Il est également nécessaire de définir l'importance du point de gravité des points aimants, en effet si on demande simplement à l'algorithme de relier les 4 points par le chemin le plus court, deux scénarios peuvent apparaitre :

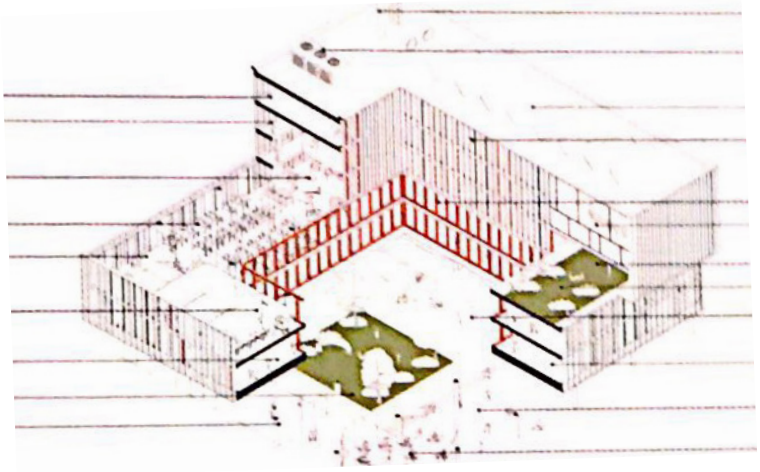


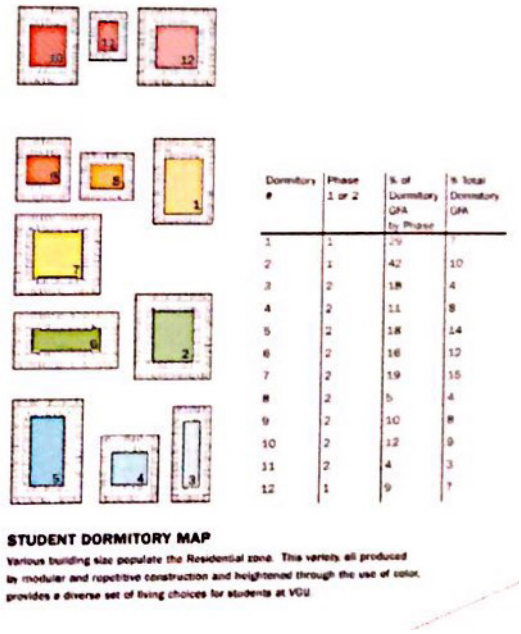


Le premier avec le centre de gravité des quatre points, le second sans. Les deux proposent différents intérêts, mais il faut pouvoir imposer l'un ou l'autre.

Ce projet utilise un second procédé très utile, celui de répéter une typologie. Le meilleur moyen d'imiter ce processus serait de créer un système de «surtypologie». Dans l'exemple du Riad, il conviendrait de créer une surtypologie « Riad » puis de demander au programme de générer plusieurs Riad adjacents. L'avantage de la surtypologie, c'est qu'elle permet de s'adapter à de nombreux cas.

2) VIETNAMESE-GERMAN UNIVERSITY CAMPUS, Ho chi minh City
Machado and Silvetti Associates





Dormitory #	Phase 1 or 2	% of Dormitory GFA by Phase	% total Dormitory GFA
1	1	29	
2	1	42	10
3	2	18	4
4	2	11	8
5	2	18	14
6	2	16	12
7	2	19	15
8	2	5	4
9	2	10	8
10	2	12	9
11	2	4	3
12	1	9	7

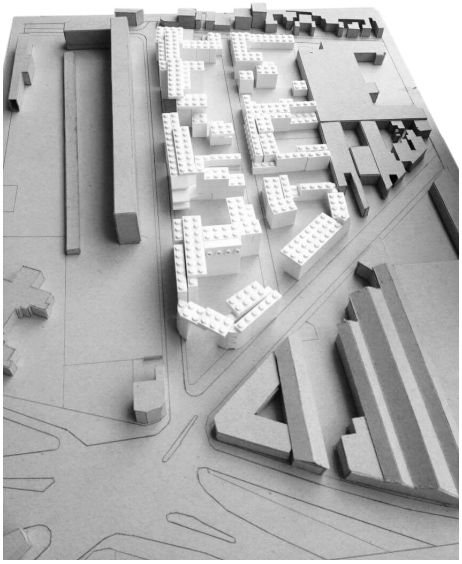
STUDENT DORMITORY MAP

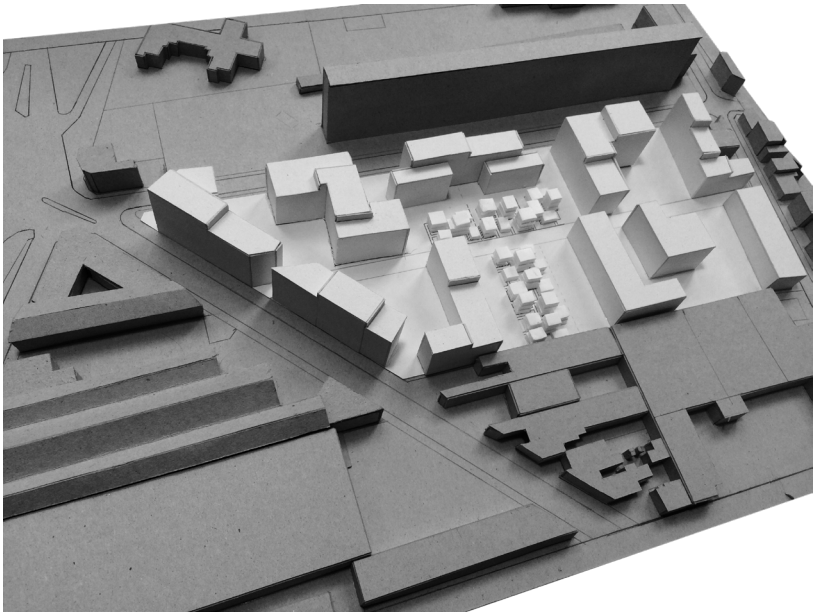
Various building size populate the Residential zone. This variety, all produced by modular and repetitive construction and heightened through the use of color, provides a diverse set of living choices for students at VGU.

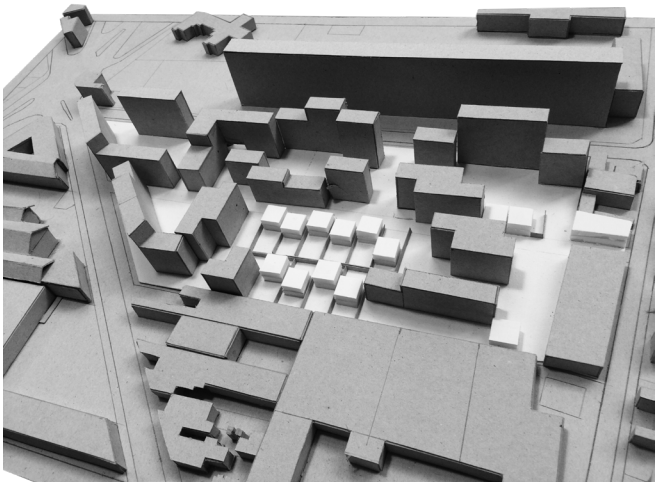
Voici un des cas pour lequel la surtypologie prend tout son sens. L'idée serait de créer plusieurs typologies de dortoirs, chacune utilisant les typologies chambres, toilettes, couloirs, etc. On pourrait ensuite incorporer ces différentes typologies de dortoirs dans le programme. On pourrait également créer une surtypologie « Unité-dortoir » comprenant juste une chambre et une salle d'eau qu'on utiliserait de façon plus libre qu'un dortoir entier.

Pour ce faire, il est nécessaire de créer une seconde interface, version simplifiée de la première permettant de dessiner ou de générer des surtypologies.

3) Projet personnel de 3e année







Ce projet de 3e année prenait place à Aubervilliers. La première phase du projet se faisait par groupes de 4, phase pendant laquelle nous définissions les grands axes de chaque groupe puis nous prenions chacun un quart du site et nous travaillons chacun de notre côté (avec un ensemble cohérent).
Ci-contre différents essais de hauteur

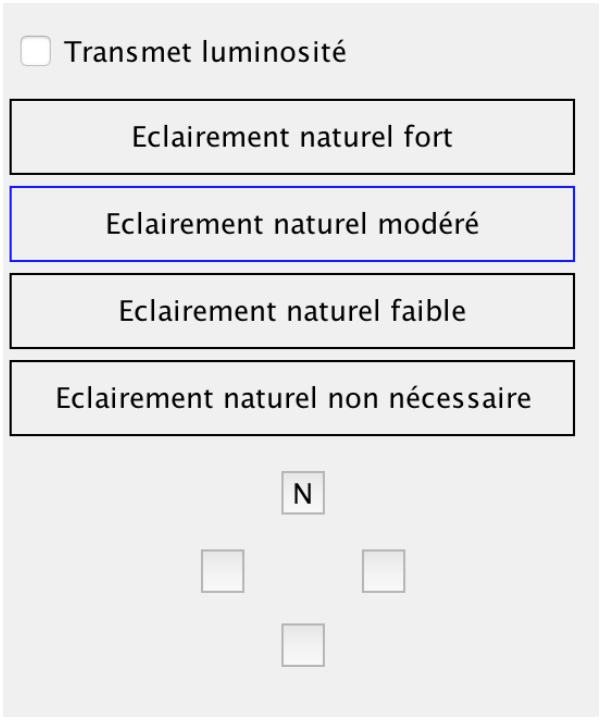
Nos recherches de groupe portaient essentiellement sur la répartition des logements dans le site ainsi qu'un travail important sur les différentes hauteurs et l'incorporation dans l'existant.
Il serait utile de pouvoir imposer un niveau de hauteur dans le programme, en proposant une nouvelle interface proposant de créer des zones de tel ou tel niveau. Cela permettrait de créer naturellement des failles, des axes ou des masses sans avoir à le spécifier par la suite. De plus, imiter le processus de création d'un projet permet de mieux comprendre les différents enjeux.

Paramètres et modélisation

Comme expliqué dans la partie concernant les bases du programme, il est important de définir des paramètres qui serviront d'une part à la génération des plans elle-même et d'autre part au calcul des scores et à confronter les différentes dispositions obtenues. Dans cette partie je vais présenter différents paramètres comme la luminosité, l'acoustique, l'énergétique, etc. cependant seuls deux d'entre eux ne sont actuellement programmés (à savoir la luminosité et le paramètre typologique) les autres ne sont que le fruit de recherches non transformées faute de temps. Malgré leur absence dans le programme final, ces paramètres m'ont néanmoins aidé dans le développement du programme.

1) Luminosité

La luminosité est sans doute le paramètre auquel on pense en premier lorsqu'il s'agit d'architecture. L'éclairement, le rapport à l'extérieur, à la chaleur, sont des enjeux qui dépassent le simple fait de pouvoir voir. Pour faire simple, j'ai divisé les besoins des typologies en quatre catégories puis j'ai suivi un modèle physique non exhaustif, mais tout de même assez réaliste correspondant au gain de lumière grâce à l'extérieur et à la perte suivant la distance.

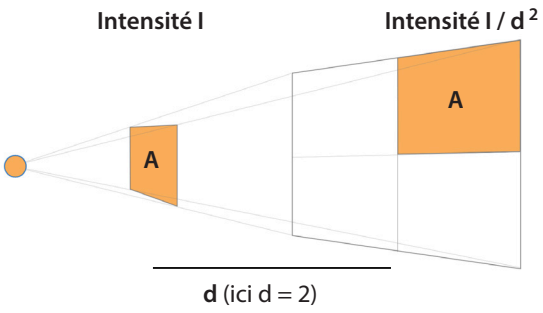


La typologie transmet-elle de la lumière aux typologies différentes ?

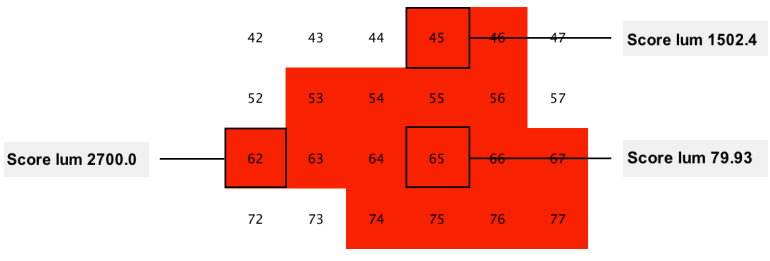
Les besoins lumineux de la typologie :
Éclairement naturel fort : 1200 - 30000 Lux
Éclairement naturel modéré : 600 - 1200 Lux
Éclairement naturel faible : 50 - 600 Lux
Éclairement non nécessaire : 0 - 50 Lux

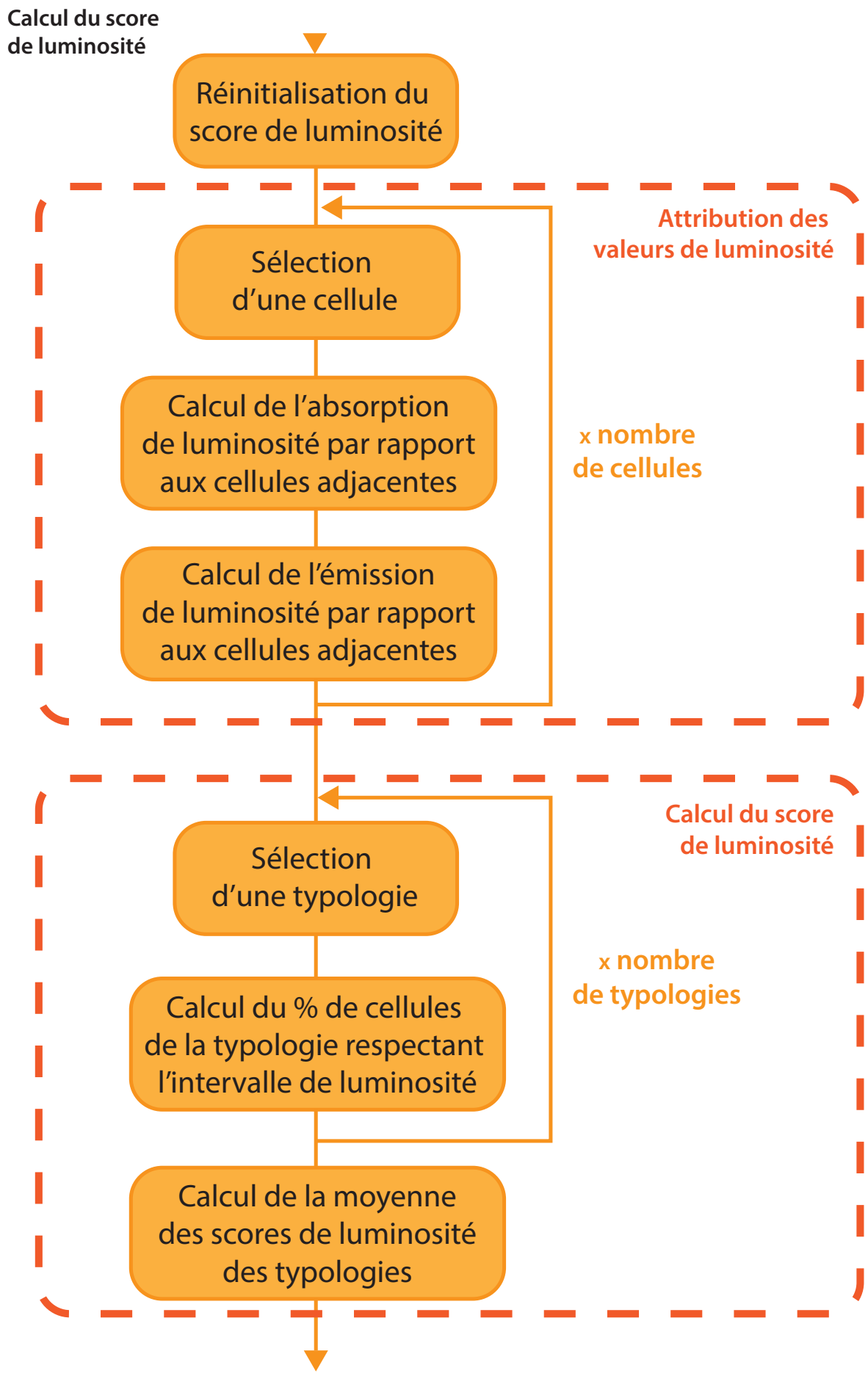
Orientation du site :
La lumière naturelle provenant du Nord est réduite à 50% de son intensité initiale, tandis que la lumière naturelle provenant de l'Ouest ou de l'Est est réduite à 75% de son intensité initiale

Modèle physique



Dans le programme




Calcul du score de luminosité
Réinitialisation du score de luminosité
Attribution des valeurs de luminosité
Sélection d'une cellule
Calcul de l'absorption de luminosité par rapport aux cellules adjacentes
Calcul de l'émission de luminosité par rapport aux cellules adjacentes
x nombre de cellules
Calcul du score de luminosité
Sélection d'une typologie
Calcul du % de cellules de la typologie respectant l'intervalle de luminosité
Calcul de la moyenne des scores de luminosité des typologies
x nombre de typologies

2) Acoustique

Le son est une vibration en milieu élastique (l'air, l'eau, mais également la matière solide à moindre échelle).
Le son est divisé en 3 catégories en fonction de sa fréquence :

- les sons basse fréquence (fréquence inférieure à 100 Hz)
- les sons communs (fréquence comprise entre 100 Hz et 2 kHz)
- les sons hautes fréquences (fréquence supérieure à 2 kHz)

Dans un bâtiment, on prend généralement en compte les fréquences comprises entre 100 Hz et 5kHz, mais l'oreille humaine peut entendre les fréquences comprises entre 20 Hz et 20 kHz.
Chaque bâtiment aura des exigences différentes, que ce soit pour se protéger du bruit extérieur ou pour ne pas gêner le voisinage.

Afin d'atténuer le bruit, il s'agit de savoir quel type de fréquence on cherche à dissiper, que ce soit en brisant l'onde, grâce aux matériaux ou en utilisant la masse du bâtiment.

Le son se mesure en décibels (dB) et sur une échelle logarithmique, ce qui implique que :

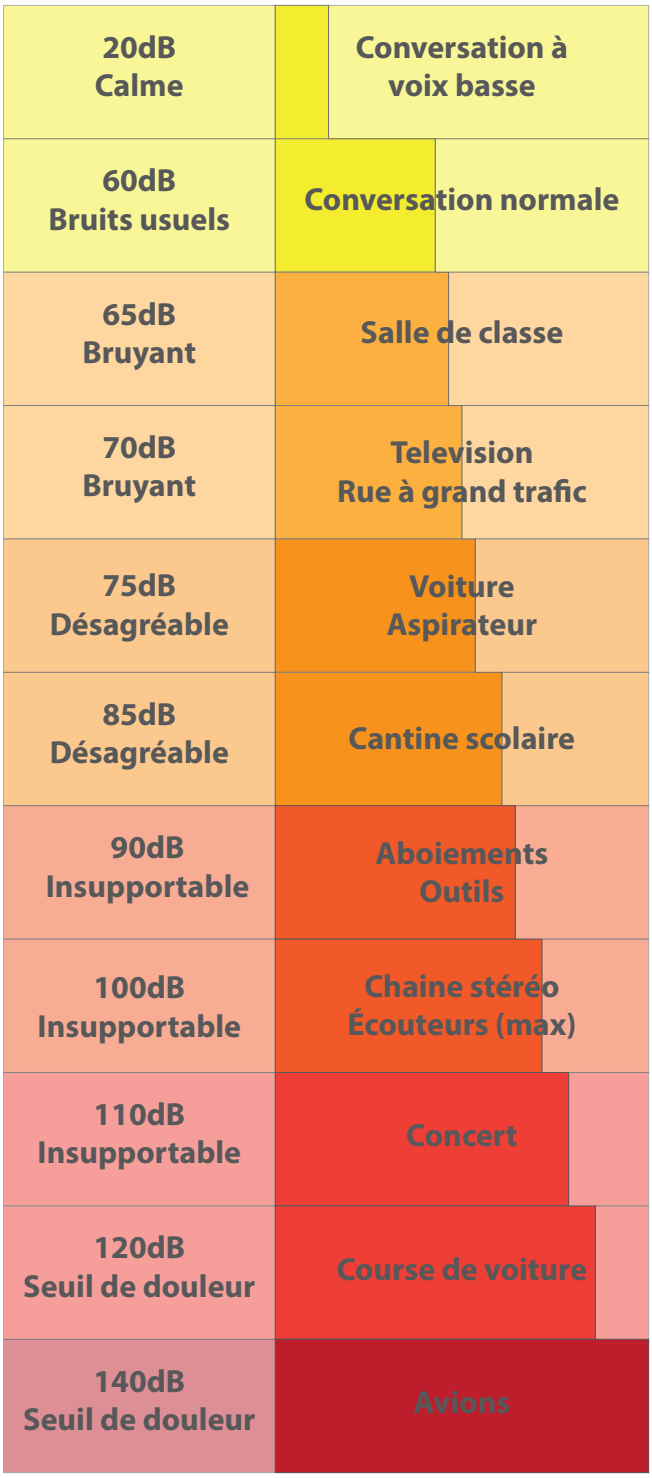
- L'addition de deux sources sonores identiques provoque une augmentation de 3dB (50db + 50db = 53 db).

- Une multiplication par 10 de la puissance sonore provoque une augmentation de 10 db (50db x 10db = 60 db)

Comment implémenter un tel modèle dans l'algorithme ?
Il conviendrait tout d'abord de voir comment le son se disperse en fonction de la distance et quelle puissance celui-ci perd. Ensuite, chaque typologie aurait une valeur d'émission et de réception d'une puissance sonore ainsi qu'une valeur de «filtre» des puissances sonores avoisinante en fonctions des autres typologies.
Par ailleurs, il faudrait pouvoir imposer des puissances sonores ponctuelles ou par zone, en effet, si l'on prend l'exemple d'un théâtre, seule la scène génère du bruit pendant la représentation.

Le modèle de dispersion généralement utilisé est celui-ci :
Dans le cas d'une source sonore à répartition sphérique, le niveau de pression du son (puissance) chute de: 20 x log(distance)
On retient généralement qu'un son perd 6dB à chaque doublement de distance en milieu libre.

On pourra alors considérer d'améliorer le filtre acoustique d'une typologie, mais cela engendrera une différence de coût.



Voici ci-contre certains exemples de sources sonores et de leur puissance.

En fonction des typologies (lieu de vie, jardin ou encore chambre à coucher) il est important d'optimiser l'utilisation de la typologie aux sources extérieures, cela évitera de perdre énormément sur le plan économique en solutions acoustiques.

Afin de considérer le paramètre acoustique dans le programme, il serait possible, comme décrit précédemment, de prendre en compte la différence de coût engendré par l'amélioration du filtre acoustique d'une typologie, mais cette solution consiste simplement à investir dans de matériaux plus performants ce qui n'aurait pas d'impact architectural réel. Il est cependant possible d'introduire une modélisation du bruit en fonction de la distance entre les typologies puis de réduire celui-ci en fonction de la masse et des matériaux traversés et de chercher à minimiser les nuisances sonores finales en fonction de la disposition des typologies. En effet, il semble évident de séparer les chambres de la rue par un petit jardin ou encore de ne pas les coller à la pièce contenant la machine à laver. Cette modélisation à l'avantage de ne prendre essentiellement en compte que la distance et par conséquent est relativement simple à mettre en place, de plus, ce qui importe ce sont les conséquences du modèle et non son exactitude.

3) Énergétique

L'idée serait également d'optimiser les pertes énergétiques grâce à l'algorithme.
De nos jours, de nombreux logiciels permettent de calculer les dépenses énergétiques de façons plus ou moins précises pour voir si un bâtiment répond aux normes imposées.

Le principe de base est très simple, comment peut-on optimiser la chaleur en hiver et à l'inverse, garder une température basse en été, et ce, de la façon la plus naturelle possible ?

Un des premiers facteurs de transfert d'énergie thermique est tout simplement la conductivité thermique inhérente aux matériaux. Voici quelques petits exemples :

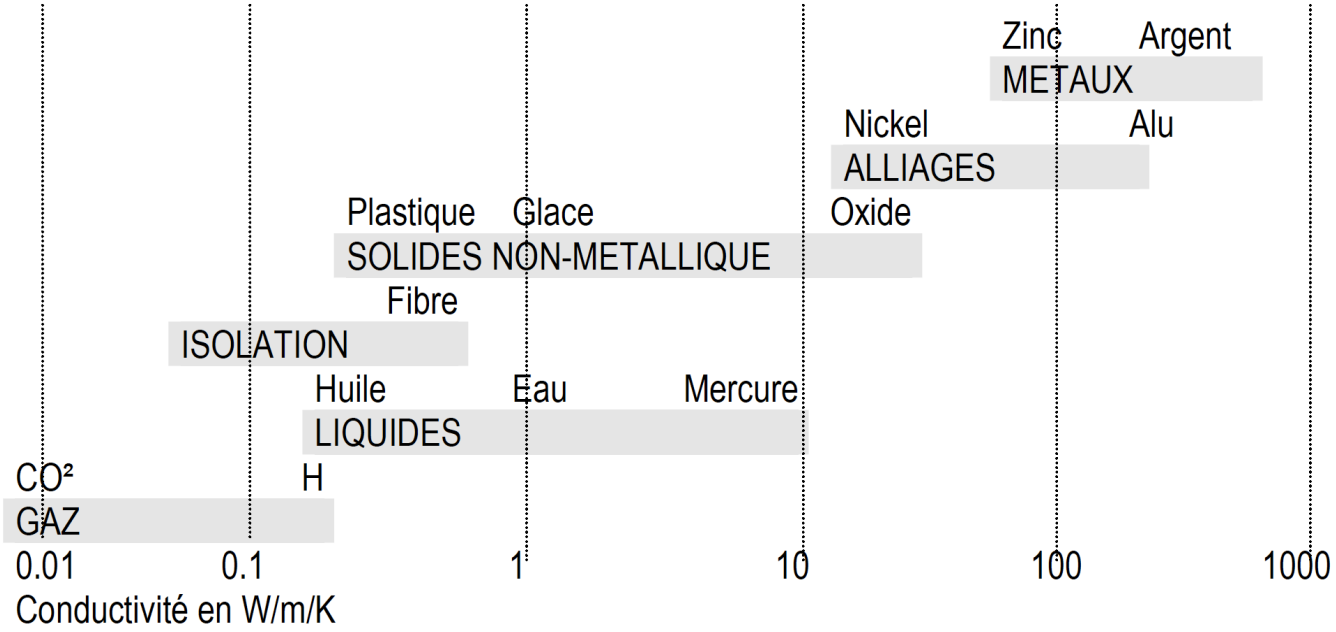


Tableau de conductivité thermique. *Source : Cours de transfert de chaleur de O. Castets (ESTP)*

Calculer avec précision quelle température il fera dans une pièce est un exercice très compliqué et dépendant d'énormément de variables (matériaux, types de régimes, types de modèle utilisé, résistances, etc.)

Il est cependant possible d'optimiser architecturalement les pertes énergiques grâce, par exemple, à l'orientation des typologies, en évitant la multiplication des contacts avec l'extérieur, etc.
Actuellement, le programme ne prend pas en compte la modélisation des déperditions énergétiques. On retrouve des parties de ce principe dans le calcul du score de luminosité notamment grâce à l'orientation, les deux étant liés, cependant l'algorithme ne calcule pas les déperditions et ne cherche pas à les optimiser. Il s'agit simplement d'évoquer la possibilité d'intégrer cette modélisation.

4) Typologiques

Le principe d'un score typologique est différent des autres paramètres pris en compte. En effet, comme nous le verrons dans la partie suivante, la génération de la grille est semi-aléatoire. Afin de gagner en efficacité, il est préférable d'orienter la génération des plans en imposant certaines dispositions tout en conservant des variables aléatoires au sein de ces dispositions. Le but n'est donc pas de vérifier si la disposition est correctement générée (le cas contraire résulterait d'une erreur de programmation), mais de distinguer les meilleures dispositions.

Pour faire prévaloir certaines grilles par rapport à d'autres, il faut comprendre le fonctionnement des plans en architecture. De façon générale (et je ne parle pas ici de vérité absolue, il existe presque autant d'architectures que d'architectes), on cherche à retrouver un «pattern», un motif ou un schéma récurrent dans un plan. Cette récurrence entraine un rythme, et à l'inverse, il est préférable d'éviter les éléments isolés, sans liens avec le reste. Pour résumer cela en un simple mot, on cherche à créer de la cohérence.

La question se pose alors, comment créer cette cohérence et comment éliminer les générations disparates ?
La réponse se trouve en statistique. Afin de mettre en valeur des tendances, les statisticiens utilisent certains outils mathématiques comme les écarts types ou la variance.
L'écart type permet de définir, par rapport à une liste de valeur, un intervalle dans lequel se trouve un pourcentage élevé de valeur (pourcentage à définir). Cet écart type nous permet justement de dégager une cohérence pour certains critères choisis à l'avance.

$$e = \sqrt{\frac{\sum | x - m |^2}{n}}$$

Avec e l'écart type, x une valeur d'une liste, m la moyenne des valeurs de la liste et n le nombre d'élément de la liste

Quels critères choisir ?

- Le nombre de typologies voisines d'une cellule : en regardant le nombre de typologies voisines différentes d'une cellule, on peut établir un lien entre les typologies. En effet, certaines typologies servent à lier différentes typologies tandis que d'autres au contraire sont isolées. En regardant le nombre de typologies voisines différentes de chaque cellule d'une typologie, on peut ainsi définir une tendance de proximité entre les typologies.

- La taille des groupes d'une typologie : l'intérêt ici est d'avoir une récurrence de taille dans une typologie. De fait, on préfère par exemple avoir des chambres de taille similaire. Ce critère permet également d'éviter les cellules isolées, ainsi on pourra favoriser le fait d'avoir deux groupes de respectivement trois et quatre cellules plutôt que deux groupes de trois et un groupes d'une seule cellule.

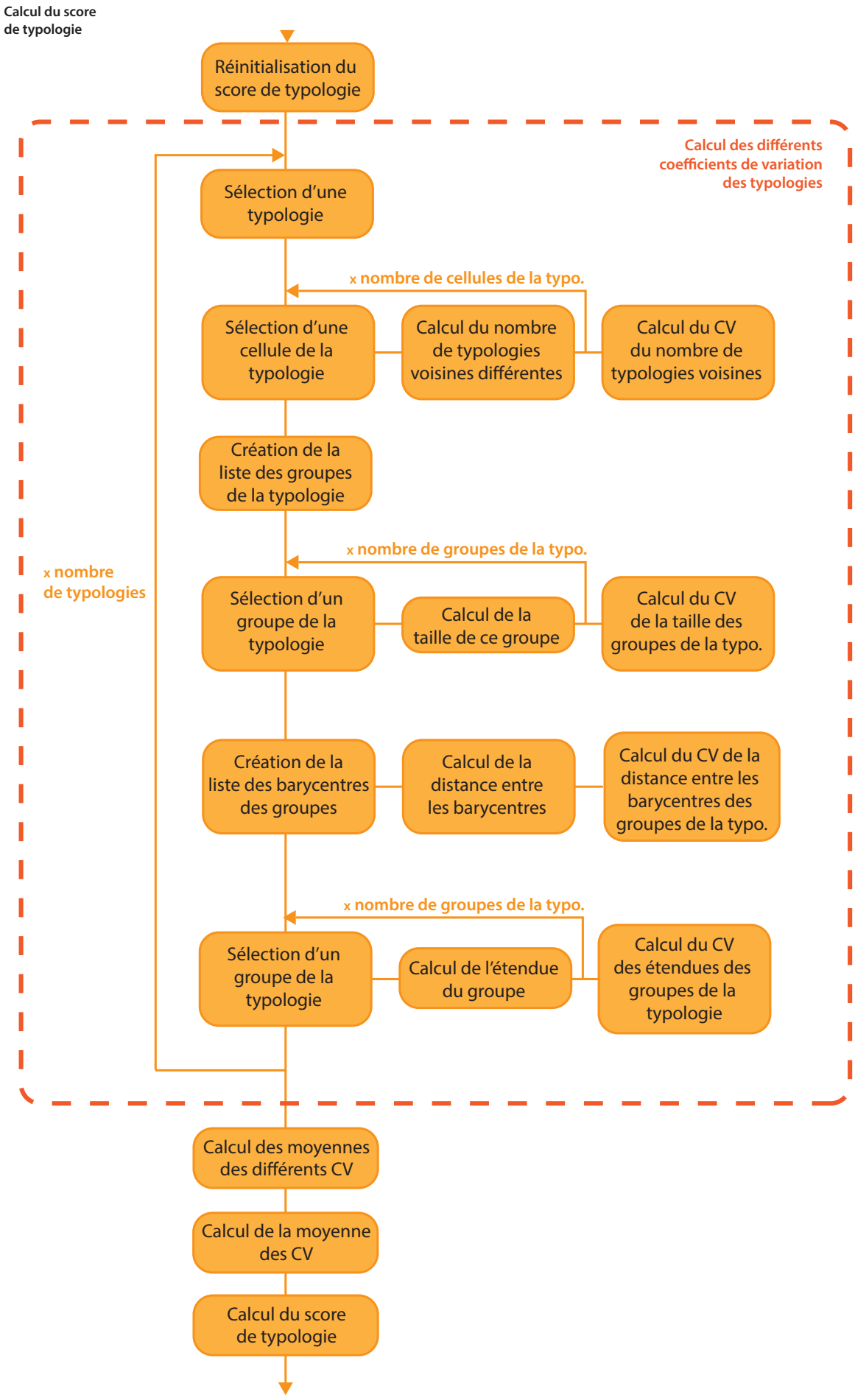
- La distance entre les groupes : de la même façon qu'il est préférable d'éviter une cellule isolée, il est également bon d'avoir une cohérence dans la disposition des groupes. Les groupes peuvent être tous proches les uns des autres, comme une «aile» d'un bâtiment ou au contraire chercher à éloigner les groupes les uns des autres, par exemple les sorties.

- L'étendue des groupes : ce critère permet de définir une similarité dans les différents groupes d'une typologie. Les groupes peuvent être concentrés ou étendus dans l'espace, une salle de classe n'a pas la même forme qu'un couloir.

Tous ces critères sont intéressants à comparer, mais sont de natures complètement différentes. Il est donc important de discrétiser les écarts types afin de pouvoir les confronter et éventuellement les coefficienter en fonction de ce que l'on recherche.

$$cv = \frac{100 \times e}{m}$$

Avec e l'écart type et m la moyenne des valeurs de la liste.


Calcul du score de typologie
Réinitialisation du score de typologie
Calcul des différents coefficients de variation des typologies
Sélection d'une typologie
x nombre de cellules de la typo.
Sélection d'une cellule de la typologie
Calcul du nombre de typologies voisines différentes
Calcul du CV du nombre de typologies voisines
Création de la liste des groupes de la typologie
x nombre de groupes de la typo.
x nombre de typologies
Sélection d'un groupe de la typologie
Calcul de la taille de ce groupe
Calcul du CV de la taille des groupes de la typo.
Création de la liste des barycentres des groupes
Calcul de la distance entre les barycentres
Calcul du CV de la distance entre les barycentres des groupes de la typo.
x nombre de groupes de la typo.
Sélection d'un groupe de la typologie
Calcul de l'étendue du groupe
Calcul du CV des étendues des groupes de la typologie
Calcul des moyennes des différents CV
Calcul de la moyenne des CV
Calcul du score de typologie

5) Économiques

L'attribution d'un paramètre économique serait relativement simple à mettre en place puisqu'il s'agit essentiellement d'un calcul métré, néanmoins celui-ci dépend de tous les autres et serait donc à établir en dernier. Par ailleurs, jamais durant mes études je n'ai réellement été confronté à des choix architecturaux liés au coût de matériaux ou de construction, bien que l'on cherche généralement l'optimisation des plans, des gains thermiques, etc.

Un critère simple serait l'attribution d'un prix au mètre carré pour chaque typologie (dépendant des matériaux, des spécifiés techniques, etc.). Par ailleurs, il serait utile d'établir un ratio superficie / périmètre d'une typologie, cela jouerait sur la disposition de celle-ci tout en introduisant une notion de coût de façade (périmètre typologie extérieure) ainsi qu'un ordre de grandeur du prix du projet.

6) Structurels

Concernant la structure, il serait judicieux d'étudier ce qu'il se fait déjà dans les logiciels utilisés par les bureaux d'études. Cependant, le but de ce programme n'est pas de calculer précisément la faisabilité d'un projet, mais plus de favoriser telle ou telle génération par rapport à des paramètres précis. On pourrait donc imposer des portées maximales lors de la génération ou simplement défavoriser par un score les portées trop petites ou trop grandes. De même, si le programme fonctionnait en 3D, il faudrait favoriser la superposition des typologies (et donc des murs porteurs) et des arrivées d'eau.

7) Normatifs

Il existe de très nombreuses normes qu'il serait plus ou moins facile d'incorporer dans le programme. Beaucoup d'entre elles utilisent la notion de distance déjà présente dans le programme, mais certaines relèvent tout simplement d'un stade plus avancé que l'esquisse de plan et ne sont donc pas pertinentes concernant ce projet.

Voici quelques normes possibles :
- Sorties incendies
- Nombres d'accès
- Largeur circulations
- Isolement coupe-feu
- (Étude d'un PLU)

Dispositions et répartition

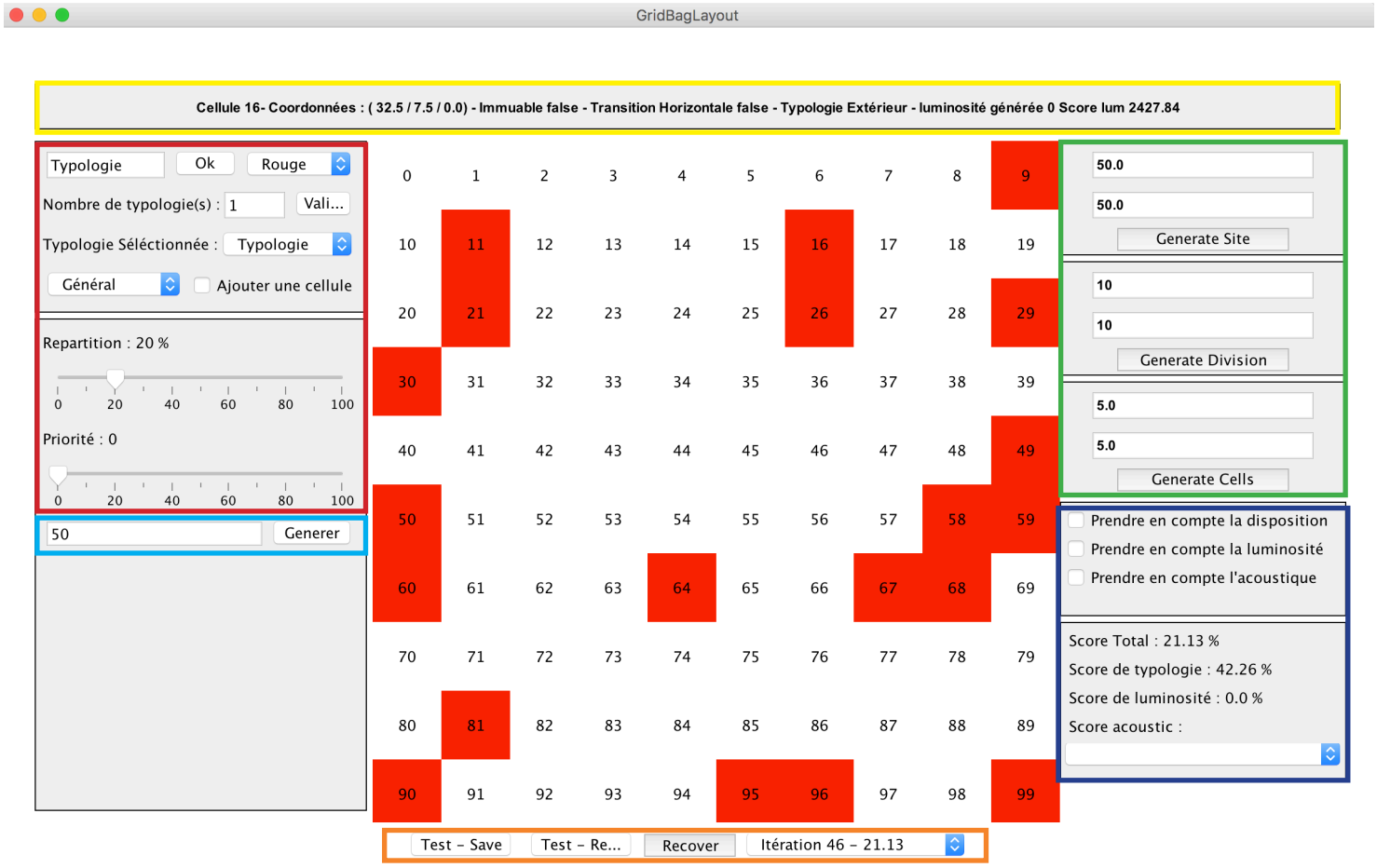
Comme précisé dans la partie concernant les bases du programme, la génération complètement aléatoire fait perdre énormément en termes de temps et d'énergie. Il est donc préférable de générer de façon semi-aléatoire.
Dans cette partie se trouvent quelques dispositions usuelles que j'ai déterminées soit via les études de cas soit en expérimentant par moi même.
Certaines dispositions ne sont pas codées, faute de temps, vous trouverez néanmoins des explications ainsi que des schémas pour comprendre ce qu'elles auraient pu devenir.

Avant de découvrir ces différentes dispositions, il est important de comprendre comment est créée l'interface visuelle. La grille est composée de cellules (cf.: les bases du programme) qui ne sont pas attribuées initialement et qui possèdent une typologie définie par défaut.
Dans un premier temps, le programme va réinitialiser la liste des cellules non attribuées, cette liste varie en fonction des options de la grille (taille, forme, cellules imposées, etc.), la réinitialisation permet de ne pas conserver de trace d'une précédente génération. La création de l'esquisse de plan se fait itérativement, en procédant typologie par typologie, le programme va donc trier les typologies par rapport aux relations typologies mères / typologies filles.
Pour chaque typologie, l'algorithme va déterminer le type de disposition puis il va attribuer les cellules sélectionnées à la liste des cellules de la typologie, et ce autant de fois qu'il y a de typologie. Une fois toutes les cellules attribuées, elles sont modifiées en fonction des paramètres de leurs typologies respectives puis la grille est rafraîchie afin de pouvoir visualiser le résultat.

La génération simple correspond à une typologie sans disposition particulière. La liste de transition correspond à la liste des cellules qui seront ensuite disposées dans la liste des cellules de la typologie. Le programme va simplement vérifier si la liste des cellules non attribuées n'est pas vide, puis choisir de façon aléatoire les cellules parmi cette liste.
Actuellement, le programme vérifie que les cellules sélectionnées ne sont pas attribuées par l'utilisateur (cellules imposées), cette façon de faire n'est pas optimale étant donné qu'il serait plus facile, plus pertinent et moins générateur d'erreurs de tout simplement retirer les cellules immuables (imposées) de la liste des cellules non attribuées dès le départ. Cette façon de faire mériterait d'être corrigée.

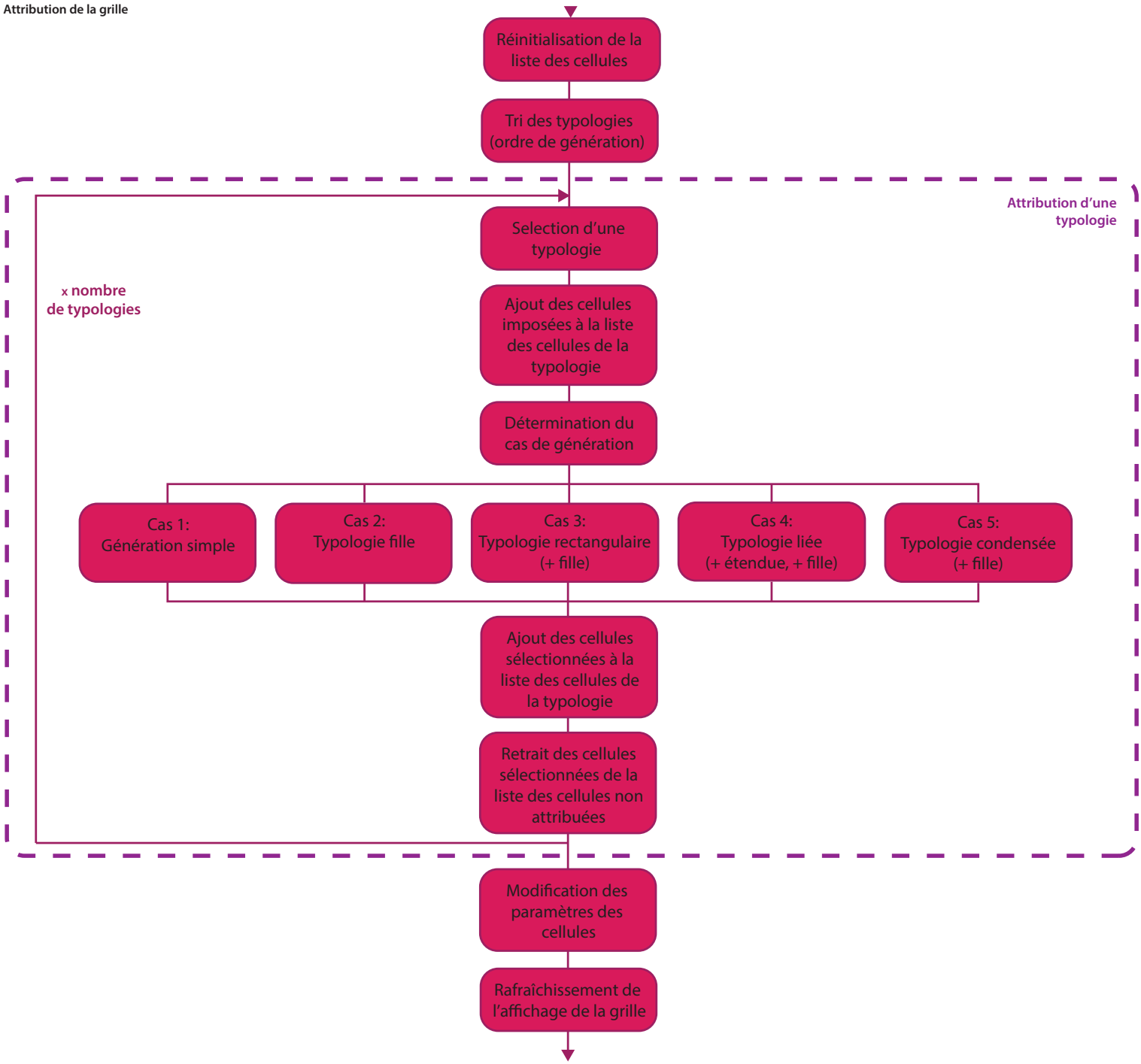
Une fois la génération effectuée, le programme va sauvegarder la disposition à l'aide d'une simple ligne de texte, en détaillant chaque cellule et en séparant ses propriétés par un «/», il sera alors capable de retrouver chaque disposition au bon vouloir de l'utilisateur.
Il serait en effet beaucoup plus simple de créer une nouvelle liste de cellules pour chaque génération, mais le système de sauvegarde par texte permet d'importer des grilles simplement en sauvegardant le fichier texte à l'extérieur du programme.

Il existe très certainement des systèmes de sauvegarde bien plus efficaces, mais celui-ci était le plus simple et le plus à ma portée lors de la création du programme.

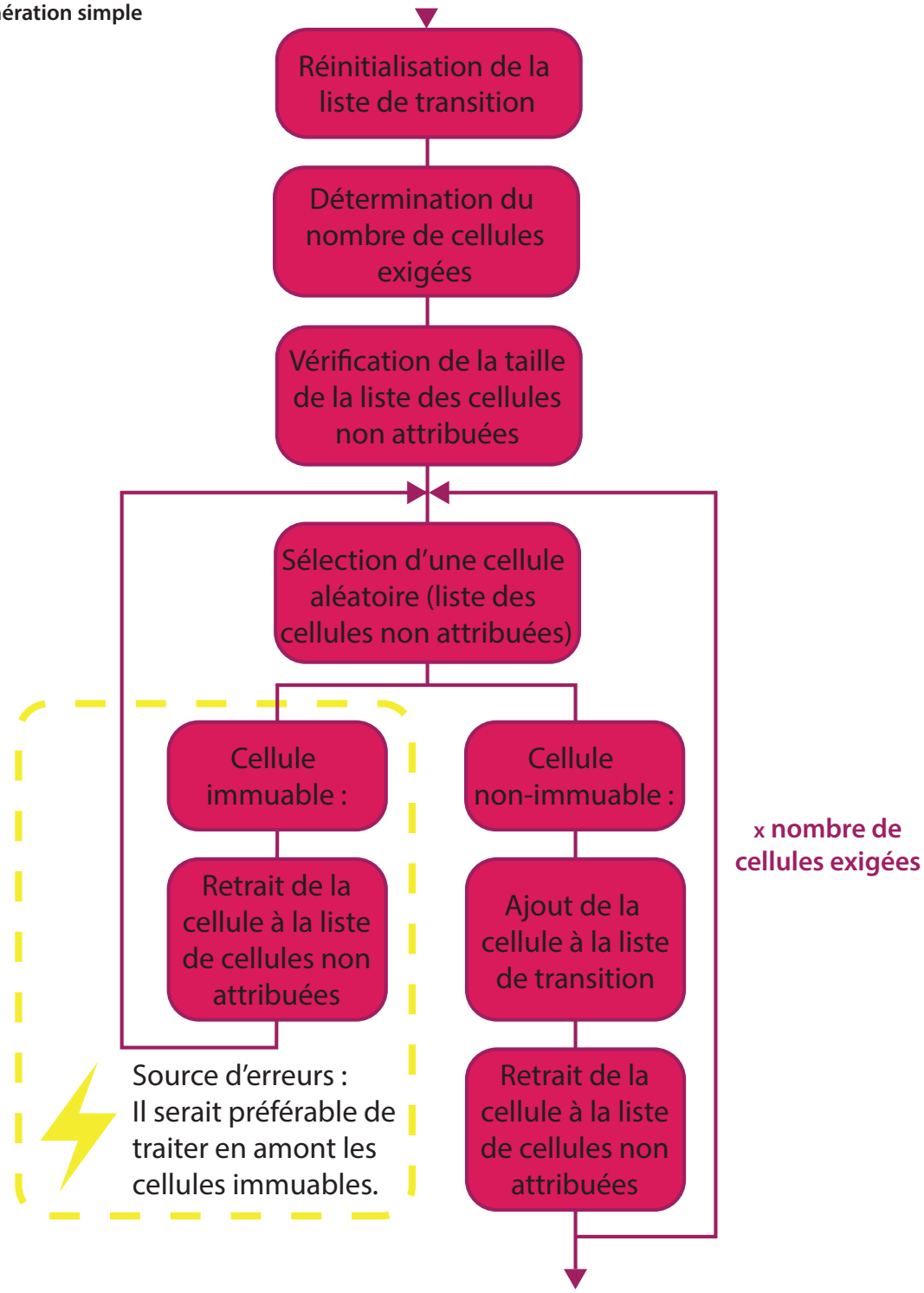


Dans l'image ci-dessus :

- Le rectangle jaune correspond à l'écran d'affichage, il donne les informations de la cellule sélectionnée
- Le rectangle rouge correspond aux typologies ainsi qu'aux différents paramètres de l'utilisateur
- Le rectangle bleu clair affiche le nombre de générations à effectuer
- Le rectangle vert affiche les paramètres de la grille (taille de la grille X/Y, nombre de cellules X/Y et taille des cellules X/Y)
- Le rectangle bleu foncé correspond aux scores de la génération
- Le rectangle orange permet de sauvegarder ou d'importer les paramètres du projet, mais également de retrouver toutes les générations effectuées classées par score
- Le centre correspond à la grille, avec en rouge, les cellules attribuées à la typologie «Typologie»


Attribution de la grille
Réinitialisation de la liste des cellules
Tri des typologies (ordre de génération)
Attribution d'une typologie
x nombre de typologies
Selection d'une typologie
Ajout des cellules imposées à la liste des cellules de la typologie
Détermination du cas de génération
Cas 1: Génération simple
Cas 2: Typologie fille
Cas 3: Typologie rectangulaire (+ fille)
Cas 4: Typologie liée (+ étendue, + fille)
Cas 5: Typologie condensée (+ fille)
Ajout des cellules sélectionnées à la liste des cellules de la typologie
Retrait des cellules sélectionnées de la liste des cellules non attribuées
Modification des paramètres des cellules
Rafraîchissement de l'affichage de la grille

Attribution de la grille :
Génération simple

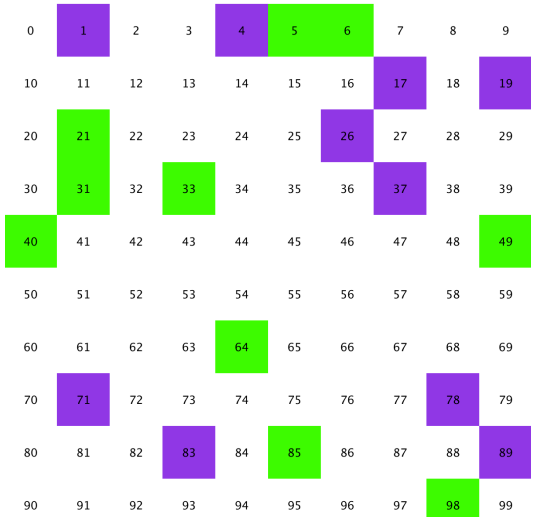


Le principe de l'attribution simple fonctionne donc sur l'aléatoire, mais on retrouve cet attribut dans chaque disposition afin de multiplier les possibilités et éviter d'obtenir des répartitions similaires ce qui fausserait les résultats.
Par exemple, dans le cas d'une disposition rectangulaire, la cellule à la base du rectangle ainsi que la direction de l'orientation du rectangle sont choisies de façon aléatoire. Il en va de même pour les autres.

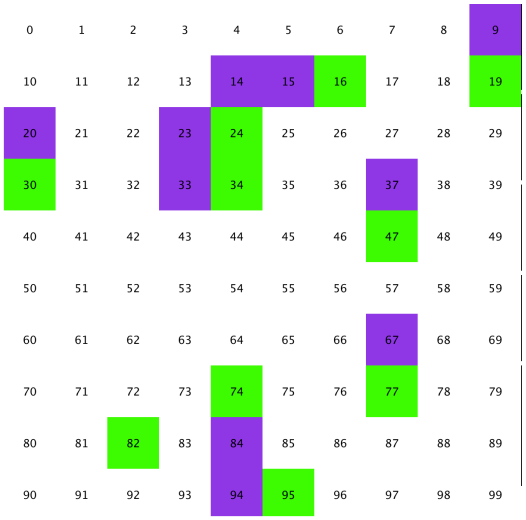
1) Typologies mères et filles

Le principe de typologie mère et de typologie fille permet d'instaurer une hiérarchie de typologie, en effet, une typologie fille sera adjacente à la typologie mère. Cela permet également de relier les typologies entre elles, une typologie de distribution comme des couloirs sera mère de plusieurs filles, elle devra donc être générée avant ses filles pour permettre l'attribution du programme.

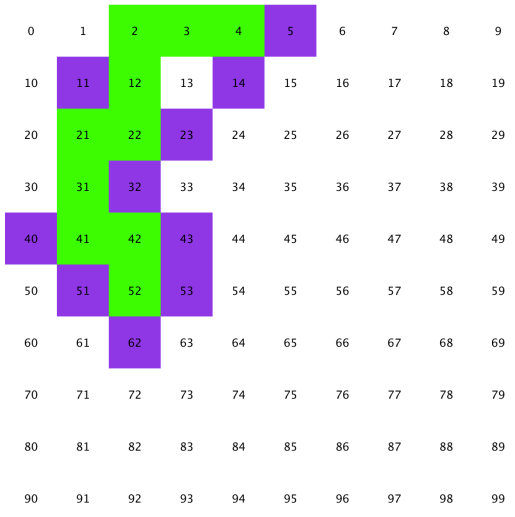
Exemple : Le couloir sera typologie mère des chambres et du salon



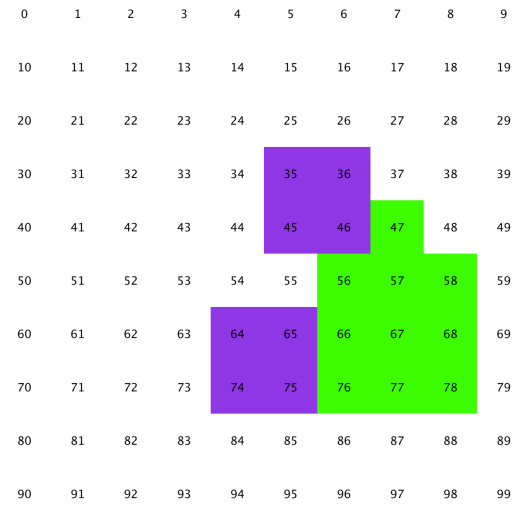
Typologie violette : répartition 10%
Typologie verte : répartition 10%



La typologie violette est fille de la typologie verte

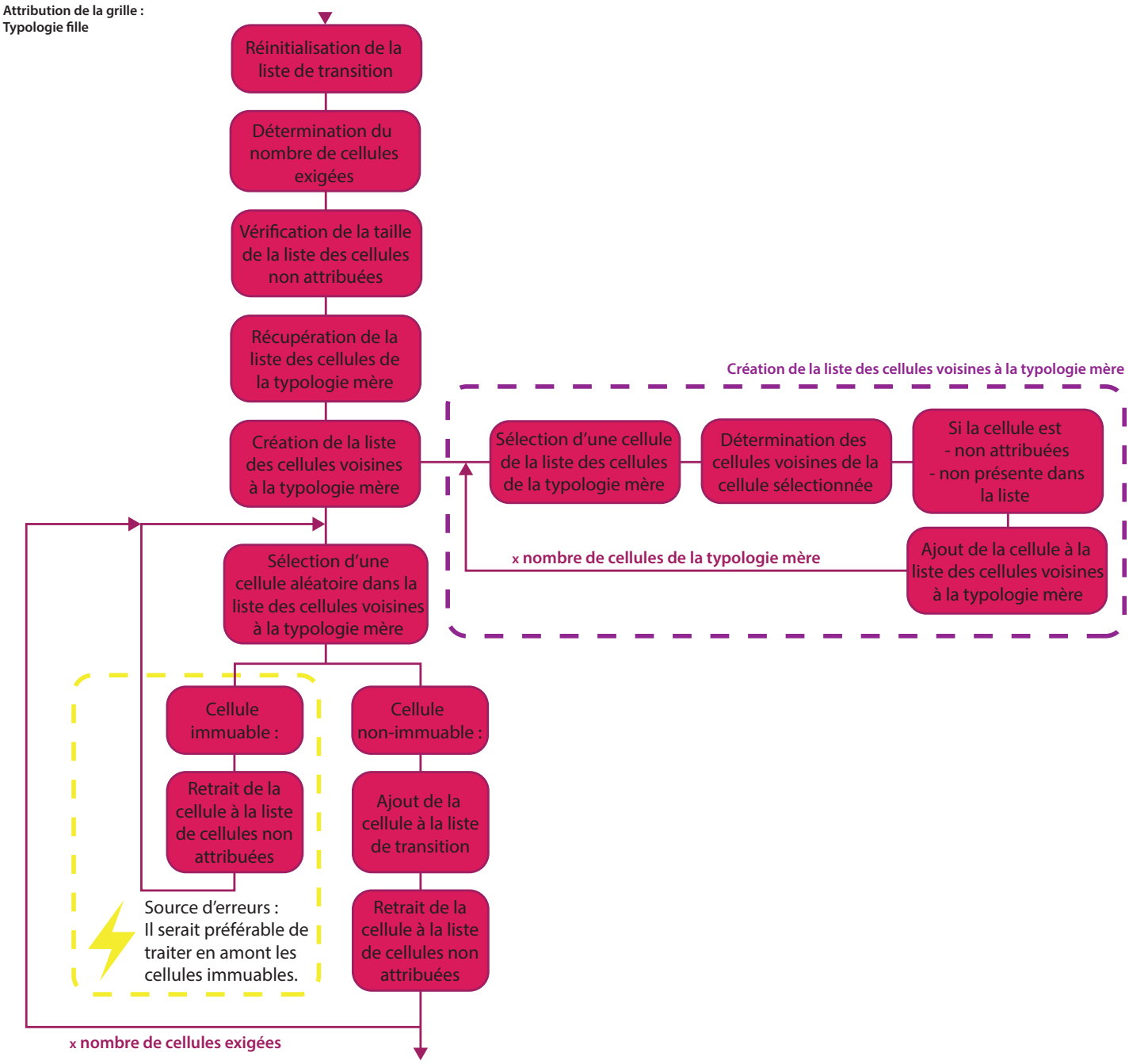


La typologie mère (verte) est liée et étendue (représentation de la circulation)



La typologie mère est condensée et la typologie fille est rectangulaire 4.

Documentation de l'algorithme :
Afin de pouvoir créer la typologie fille, il est important de hiérarchiser les typologies entre elles et créer d'abord la typologie mère en ordonnant la liste des typologies.
Le programme va choisir une cellule aléatoire dont une des cellules voisines appartient à la typologie mère et répéter ce processus autant de fois que nécessaire.
Dans le futur, il serait intéressant d'ajouter d'autres paramètres comme attribuer un nombre défini de cellules filles à une cellule mère. *Exemple : Chaque chambre possède une unique salle d'eau dédiée.*

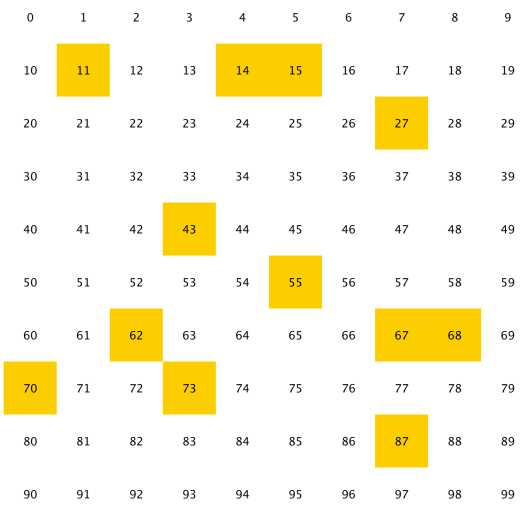


2) Rectangulaire et adaptée

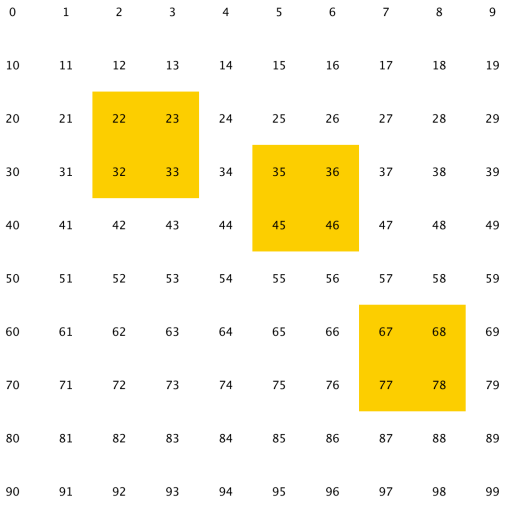
La disposition rectangulaire représente la plupart des typologies, elle permet de répartir de façon rectangulaire une typologie tout en optimisant le ratio longueur / largeur. Elle permet également de s'adapter aux cases imposées par l'utilisateur. Par ailleurs, il est également possible d'assigner la fonction typologie fille avec cette disposition.

Case(s) élémentaire(s) : 1 Ok

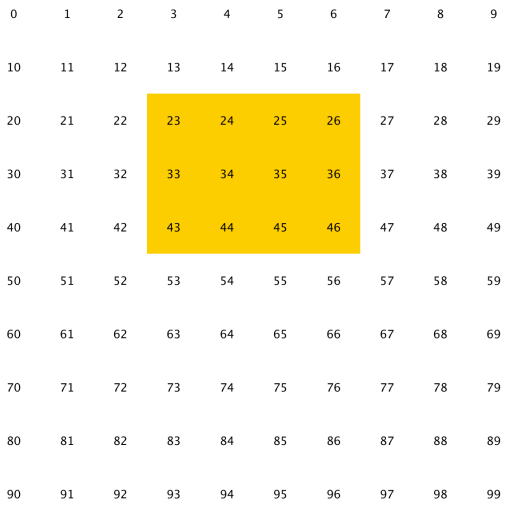
Nombre de cases élémentaires de la typologie
Exemple : Une chambre sera composée de 4 cases élémentaires de 1,5 m x 1,5 m



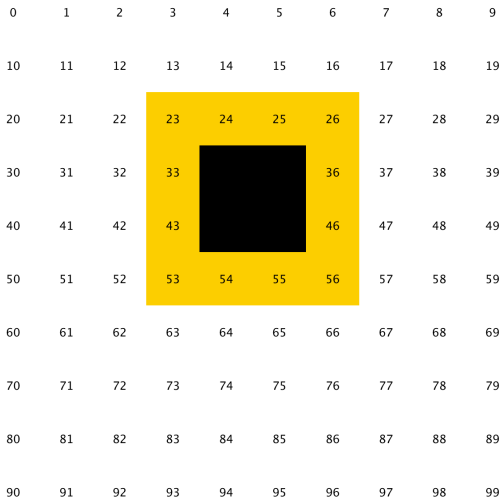
Typologie de répartition 12%



4 cases élémentaires



12 cases élémentaires

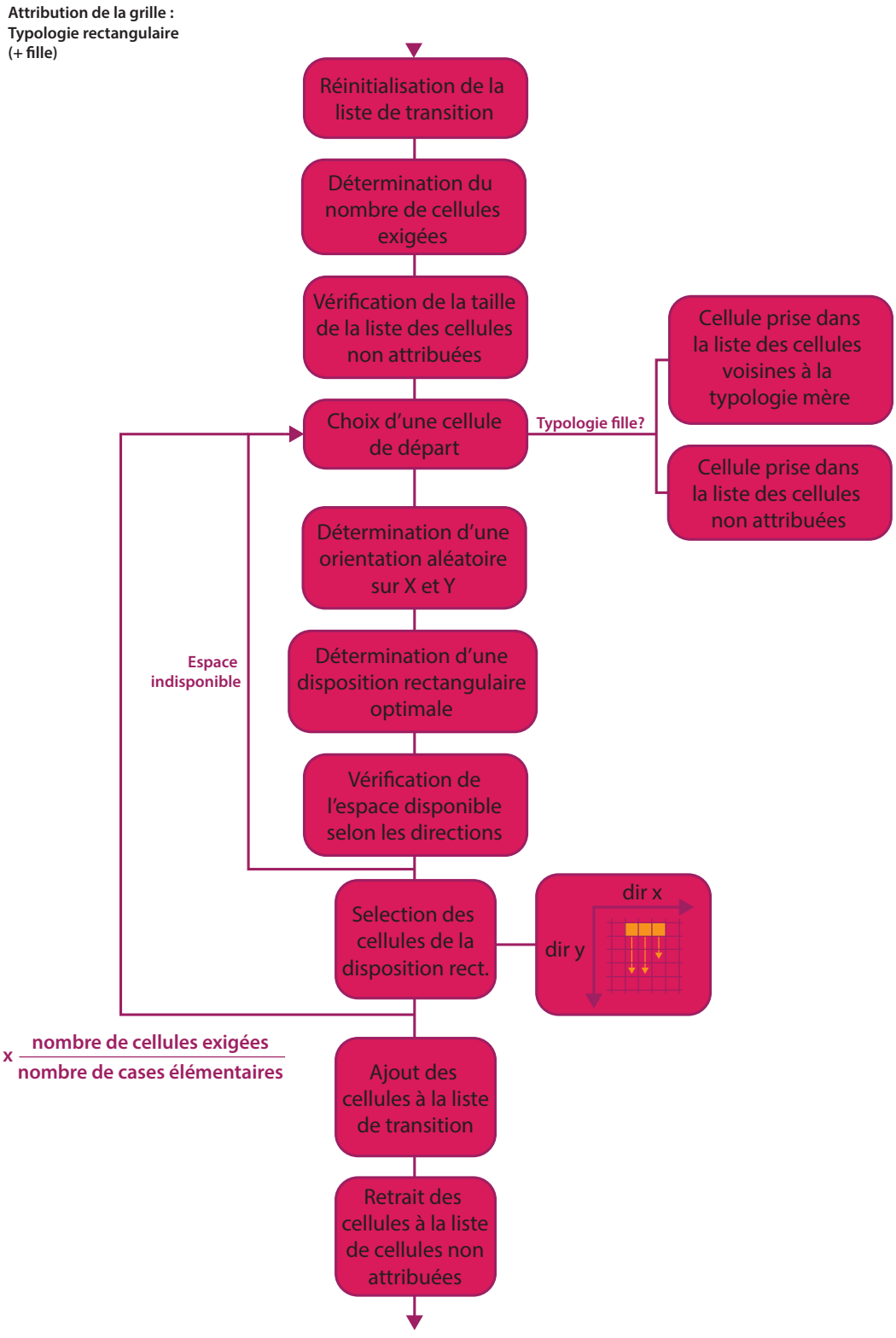


Le caractère rectangulaire s'adapte à l'existant

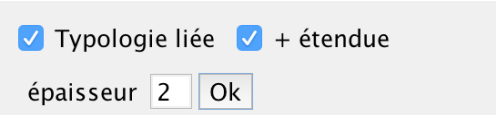
Documentation de l'algorithme :

Le programme définit une case de départ. Afin de conserver une composante aléatoire, l'algorithme choisit aléatoirement une orientation sur X et sur Y et par ailleurs il trouve la répartition rectangulaire la plus naturelle (par exemple ici, 12 = 4 x 3 et non 6 x 2).

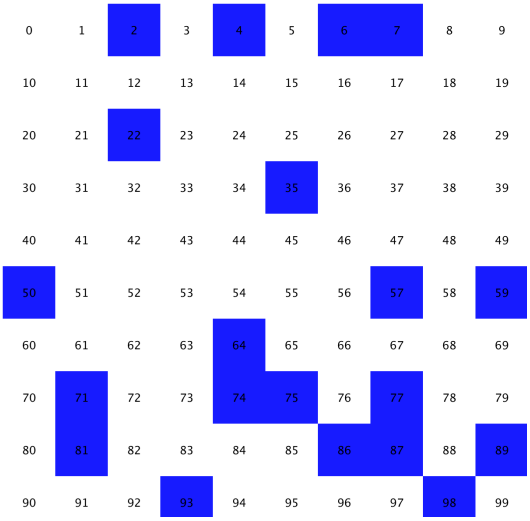
Il est important de vérifier si il existe suffisamment de place pour éviter de sortir de la grille. De même si l'algorithme rencontre un obstacle (typologie imposée par exemple) il va rajouter le nombre de cases obstacles à la suite du rectangle en ligne ou colonne, si possible, afin de respecter au mieux les paramètres de l'utilisateur.



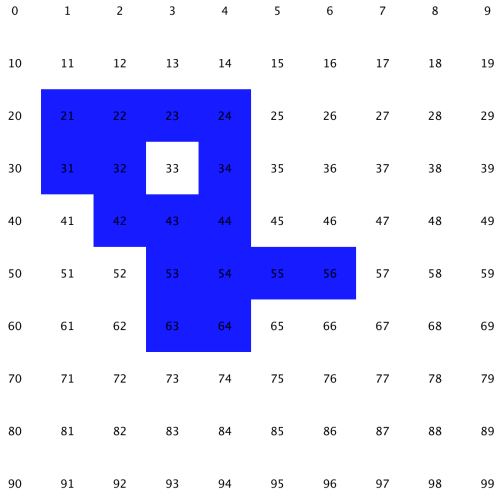
3) Liée et étendue



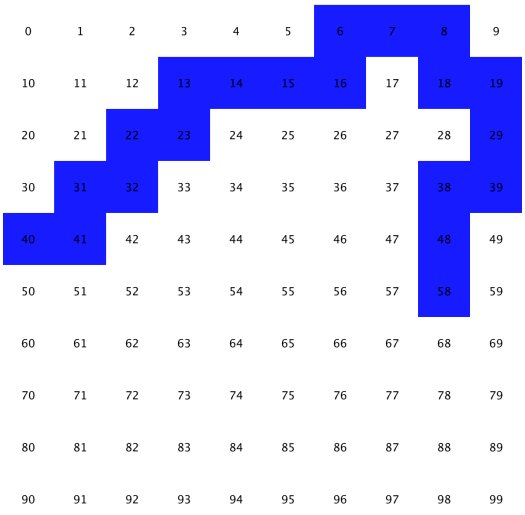
La typologie liée permet de définir un lien entre toutes les cellules. L'épaisseur représente le nombre de cellule minimal dans la largeur de ce lien. *Exemple : distribution horizontale*



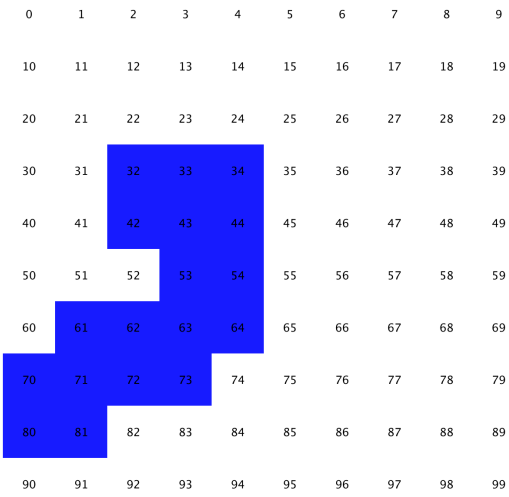
Typologie de répartition 20%



Typologie liée



Typologie liée et étendue



Typologie liée et étendue d'épaisseur 2

Documentation de l'algorithme :

Le programme définit une case de départ et attribue la typologie à une cellule voisine aléatoire par itération. Si aucune cellule voisine n'est disponible, il repart d'une cellule précédemment attribuée aléatoirement.

Dans le cas d'une typologie étendue, lorsque le programme choisit une cellule voisine, il va sélectionner une cellule qui elle même n'est pas voisine à une autre cellule de la même typologie (sans tenir compte de la cellule de l'itération). L'idéal serait de pouvoir définir un aimant de direction.

Pour tenir compte de l'épaisseur, le programme va étudier la direction prise par la typologie, si celle-ci part vers l'Est ou l'Ouest, l'épaisseur se fera sur une direction choisit aléatoirement entre le Nord et le Sud, et inversement. De même, lorsque la typologie «tourne» dans le sens opposé de l'épaisseur, la nouvelle épaisseur est définie sur le sens de l'ancienne cellule, afin d'éviter un dédoublement de l'épaisseur.

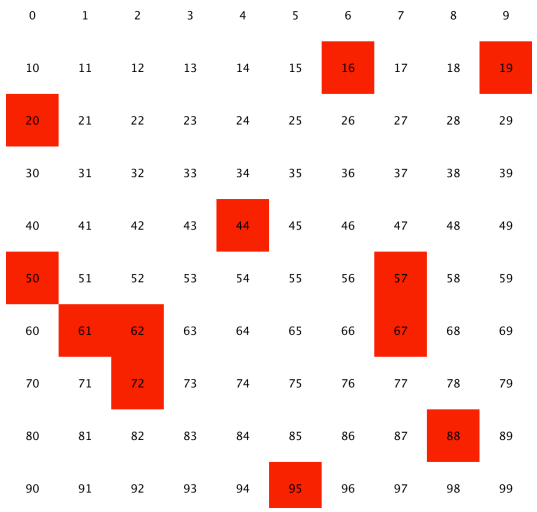
Attribution de la grille :
Typologie liée
(+ étendue)
(+ fille)

4) Condensée et points d'attraction

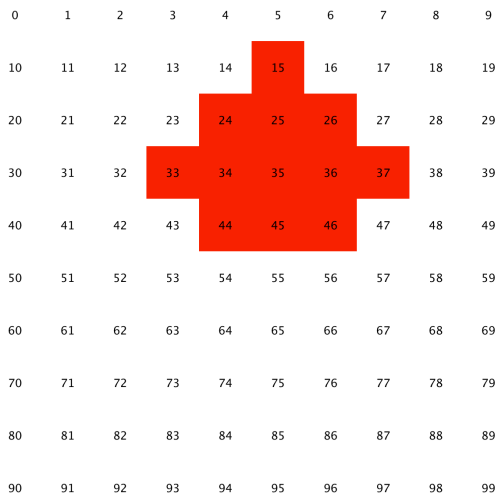
La typologie condensée permet, comme son nom l'indique, de condenser les cellules d'une typologie autour d'un point précis. Ce caractère permet également de s'adapter aux cellules immuables en simplement triant les cellules par proximité à la cellule choisie. Ainsi la condensation peut se faire en n'importe quel point de la grille, même un coin, cela ne pose pas de problème. Pour l'instant, le point de condensation est choisi aléatoirement dans la grille (ou parmi les cellules filles), il serait cependant intéressant de pouvoir le préciser manuellement.

☑ Typologie condensée

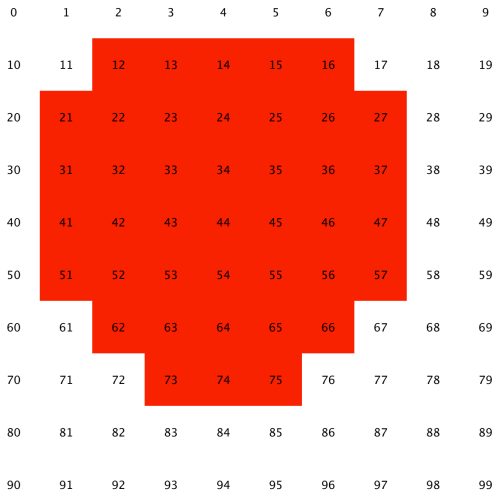
Nombre de cases élémentaires de la typologie
Exemple : Une chambre sera composée de 4 cases élémentaires de 1,5 m x 1,5 m



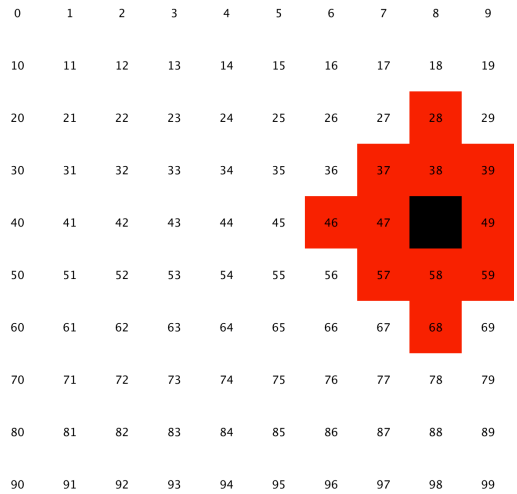
Typologie de répartition 12%



Cellule aimant : 35

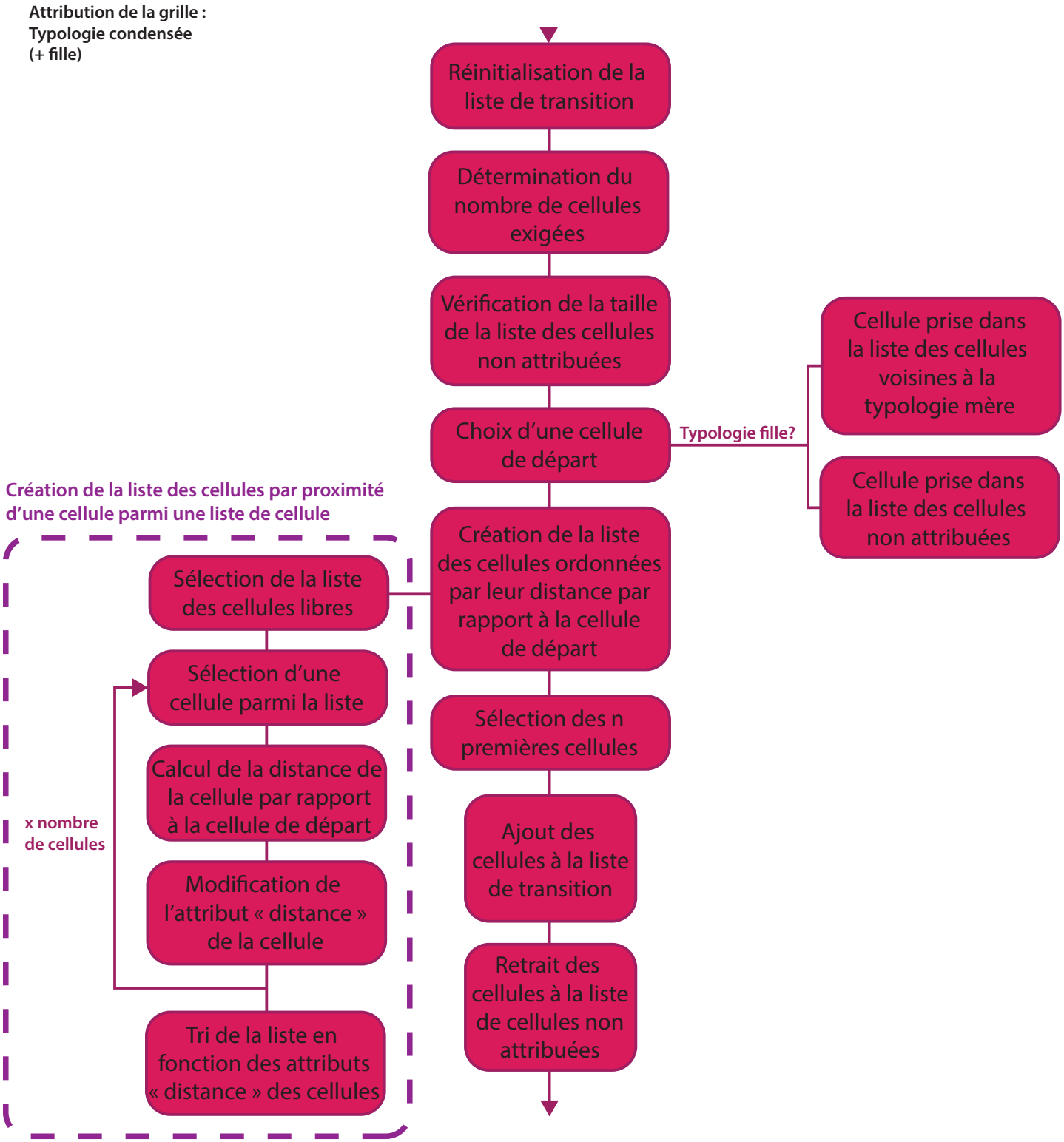


Répartition : 41%
Cellule aimant : 44



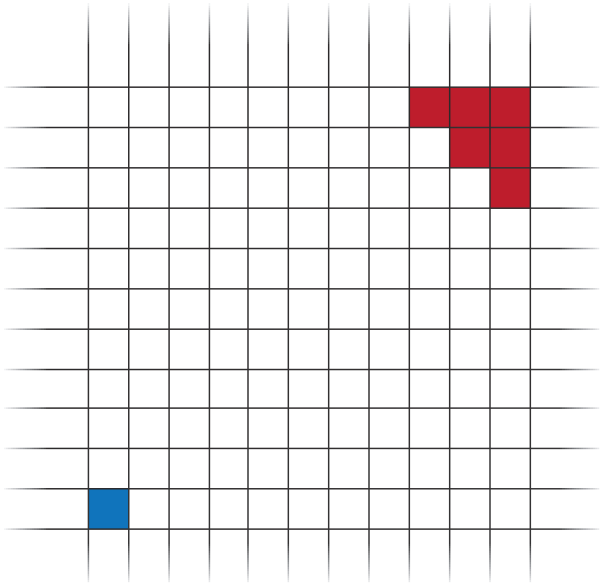
Le caractère condensé s'adapte à l'existant

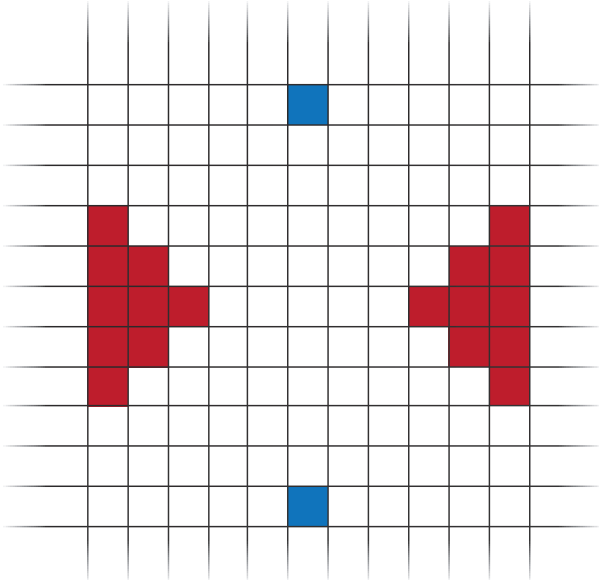
Documentation de l'algorithme :
Le programme défini une case de départ, pour l'instant celle-ci est aléatoire, mais le but est de rajouter si possible une case imposée ou plusieurs avec possibilité de pondérer les «aimants».
Le programme va ordonner les autres cellules par valeur de proximité avec la cellule de départ. Si les cellules ne sont pas occupées alors elles sont assignées à la typologie.

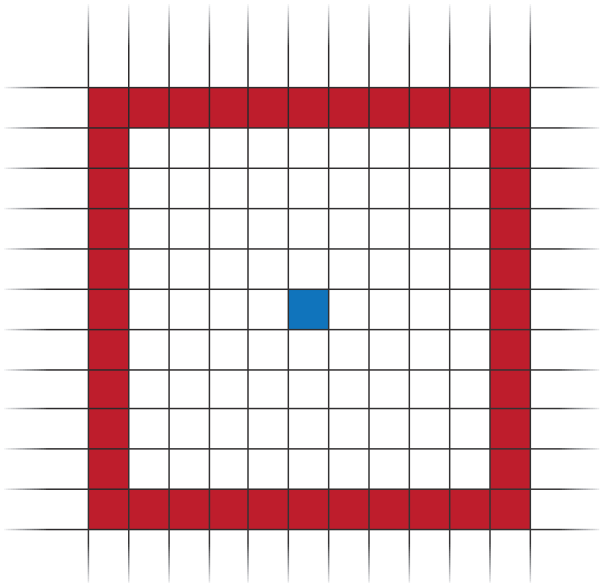


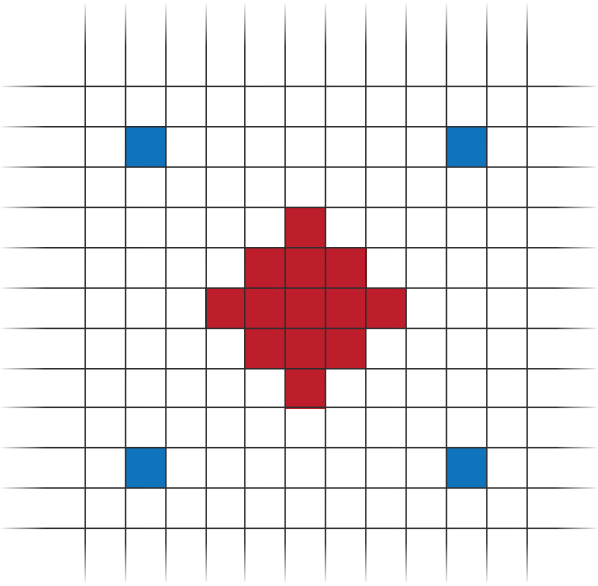
5) Répulsive

Non implanté actuellement.
Le principe de la répartition répulsive est l'inverse de la répartition condensée.
L'idée est d'ordonner les cellules par éloignement d'une cellule (ou d'une zone de cellule) et de choisir les plus éloignées.
On pourrait également choisir une typologie de répulsion en effectuant simplement la distance moyenne d'une cellule à chaque cellule de cette typologie ou en créant un centre de gravité de la typologie de répulsion.



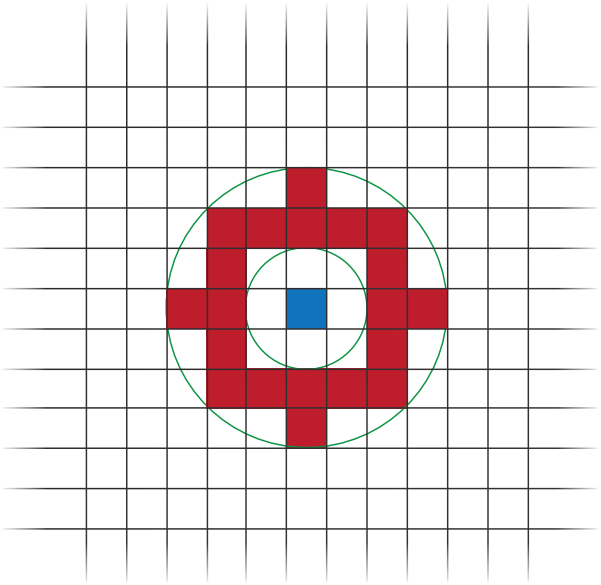


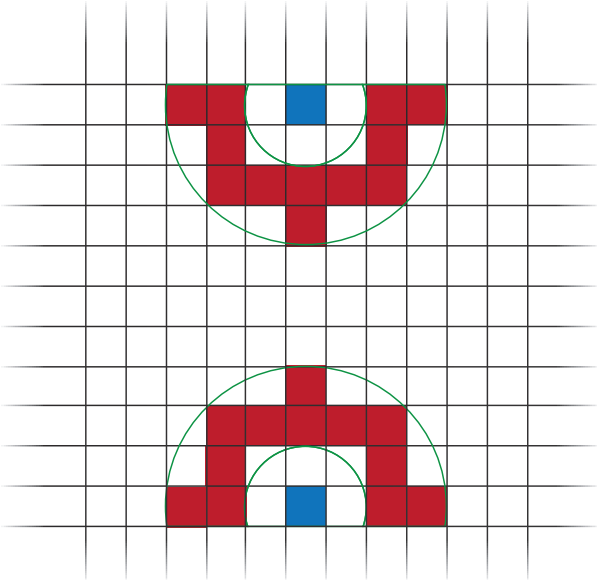


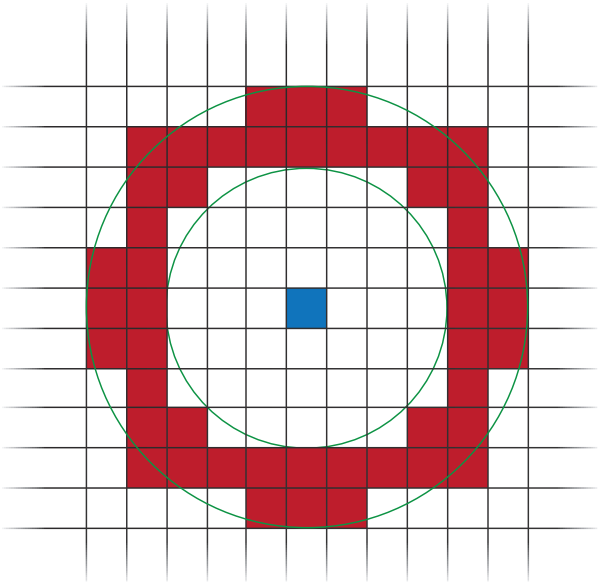


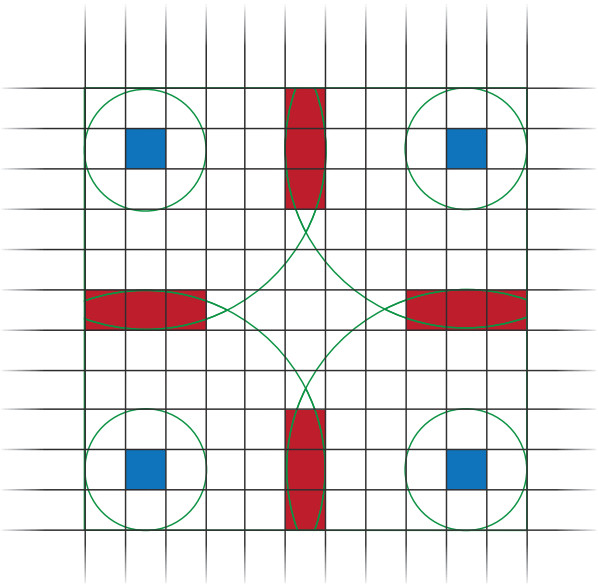
6) Encadrée

Non implanté actuellement.
Il peut être intéressant d'encadrer la typologie entre deux distances définies par rapport à un point ou une autre typologie.
Pour ce faire, l'idée est de simplement ne pas prendre en compte les cellules trop éloignées ou trop proches et de répartir la typologie dans le groupe de cellules restantes.



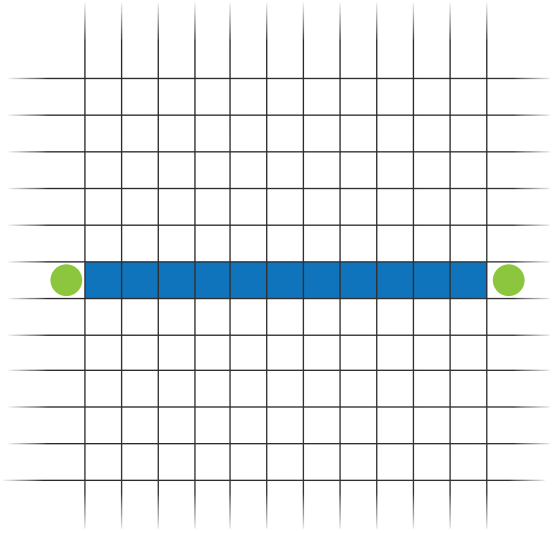




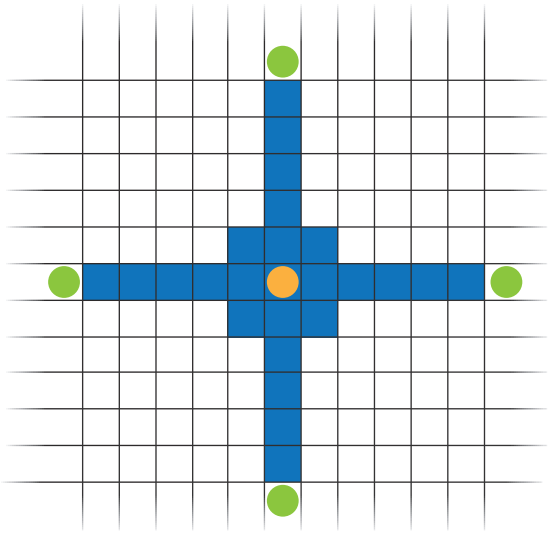


7) Jonction

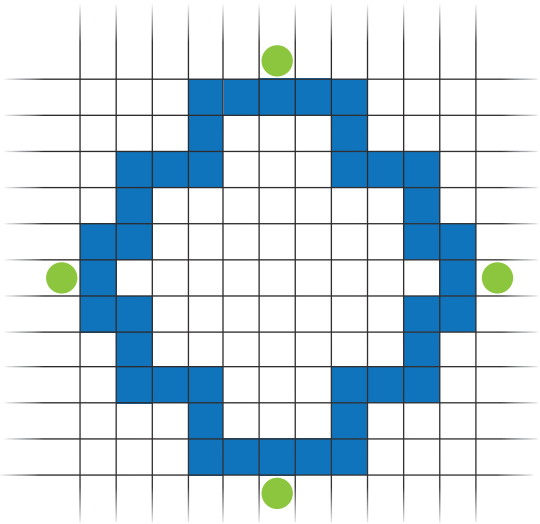
Non implanté actuellement.
Une typologie de jonction permettrait de joindre différents objets de nature différente. Par exemple, il serait possible de joindre un point A et un point B ou encore deux typologies différentes, soit par le chemin le plus court, soit par des points imposés soit par centre de gravité.



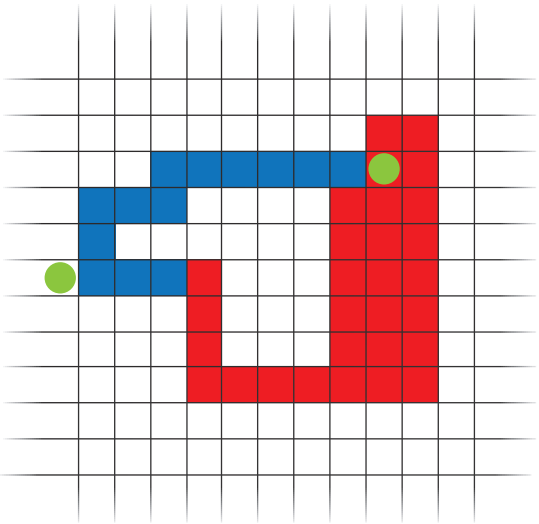
Jonction points A et B



Jonction 4 points avec centre de gravité

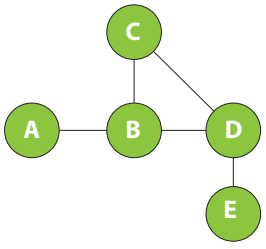


Jonction 4 points sans centre de gravité



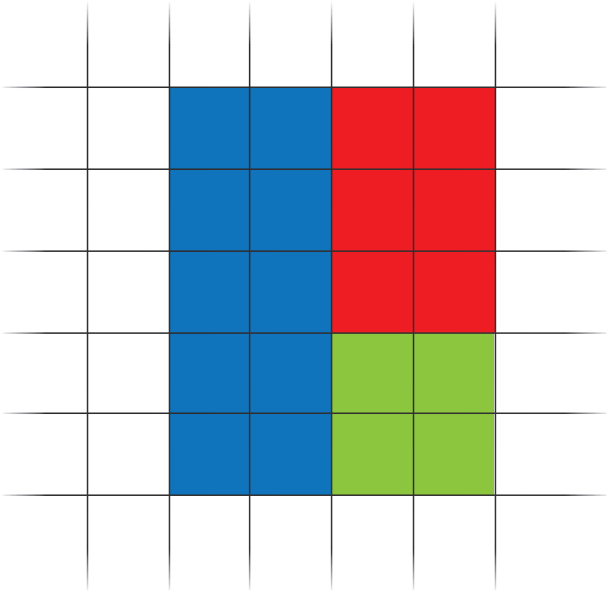
Jonction point A et typologie simple
Jonction point A et typologie point imposé

Il serait également utile de permettre la possibilité de définir l'épaisseur de la jonction ou encore de créer une interface de maillage comme ci-contre :

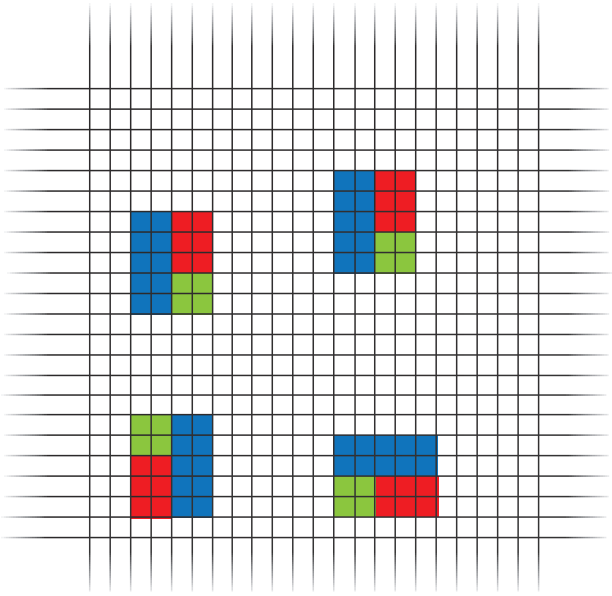


8) Surtypologie

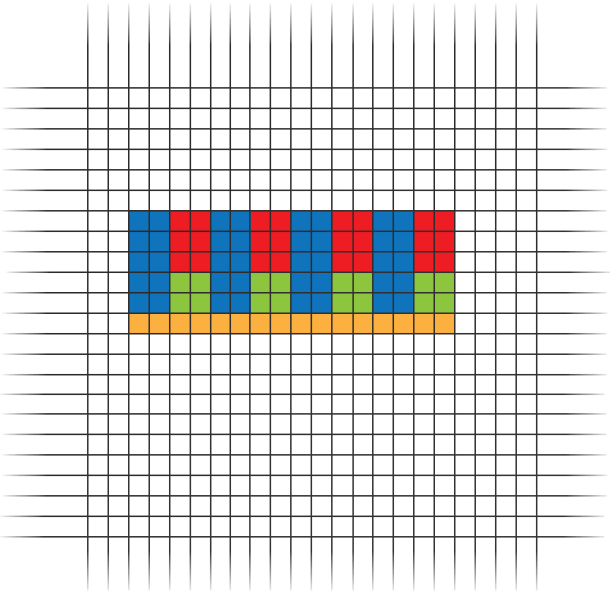
Non implanté actuellement.
La surtypologie permet de créer des unités typologiques tout en utilisant les typologies existantes et les différentes méthodes de génération.



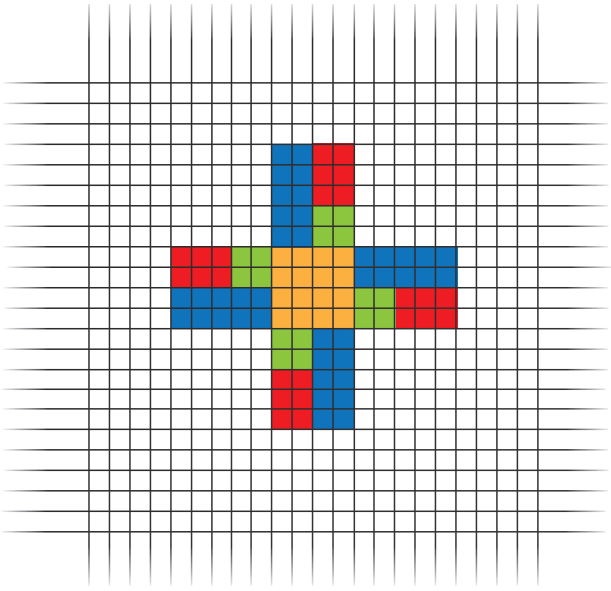
Unité de surtypologie



Génération aléatoire



Surtypologie avec typologie mère



Surtypologie condensée

9) Mélanges de dispositions

Chaque répartition ou disposition doit pouvoir être associée à certaines autres. Par exemple, une typologie fille peut être également liée et étendue, de même, une typologie encadrée doit pouvoir être fille d'une autre typologie, il faudra alors voir quelles cellules répondent aux deux critères.
La question qui se pose dans ce cas est : comment hiérarchiser les paramètres ?
Vaut-il mieux privilégier le nombre de cellules ou les critères de répartitions ? Et quels critères sont prioritaires ? Cela dépend certainement de la typologie, certaines sont normées d'autres moins.

Expérimentation

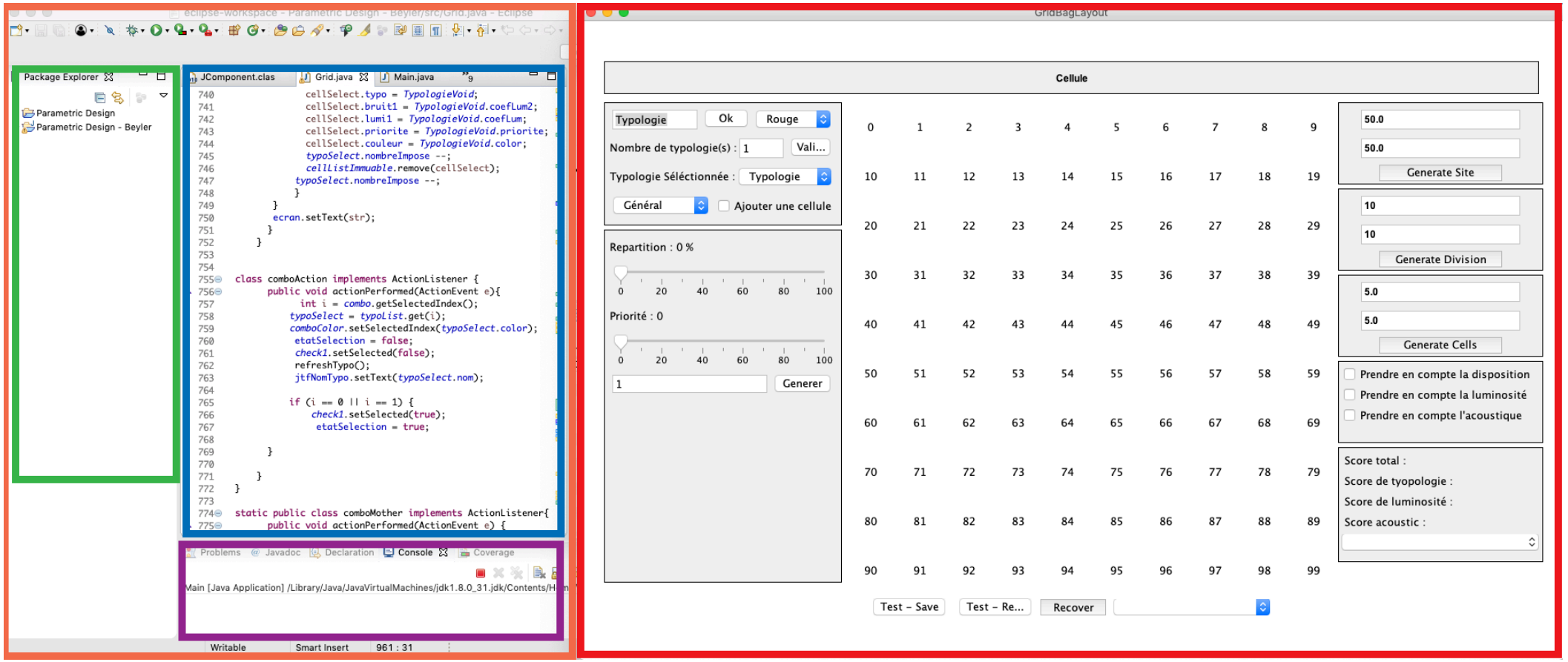
Dans cette partie nous allons étudier en détail le fonctionnement du programme. Cette expérimentation à plusieurs objectifs bien distincts : dans un premier temps, elle permettra tout simplement d'obtenir des résultats et ainsi d'observer la pratique appliquée à la théorie; dans un deuxième temps, cela permet au lecteur d'appréhender par lui même les différents algorithmes et de clarifier d'éventuelles interrogations; et enfin dans un dernier temps, on pourra étudier les limites actuelles du programme et anticiper de potentielles améliorations.
Il faut avant tout se figurer qu'actuellement la génération d'esquisses de plan présentée dans ce mémoire n'est absolument pas terminée et encore moins optimisée, mes compétences en programmation, informatique et même architecture sont limitées et il serait illusoire de prétendre obtenir des résultats satisfaisants, il s'agit néanmoins là d'une ouverture au sujet qui, je l'espère, conduira à d'hypothétiques futurs travaux.

Tout d'abord, nous verrons la mise en place du programme ainsi que de divers préparatifs essentiels à cette expérimentation, ensuite nous observerons le fonctionnement de l'algorithme pas à pas et enfin nous analyserons les résultats obtenus.

1) Mise en place de l'expérimentation

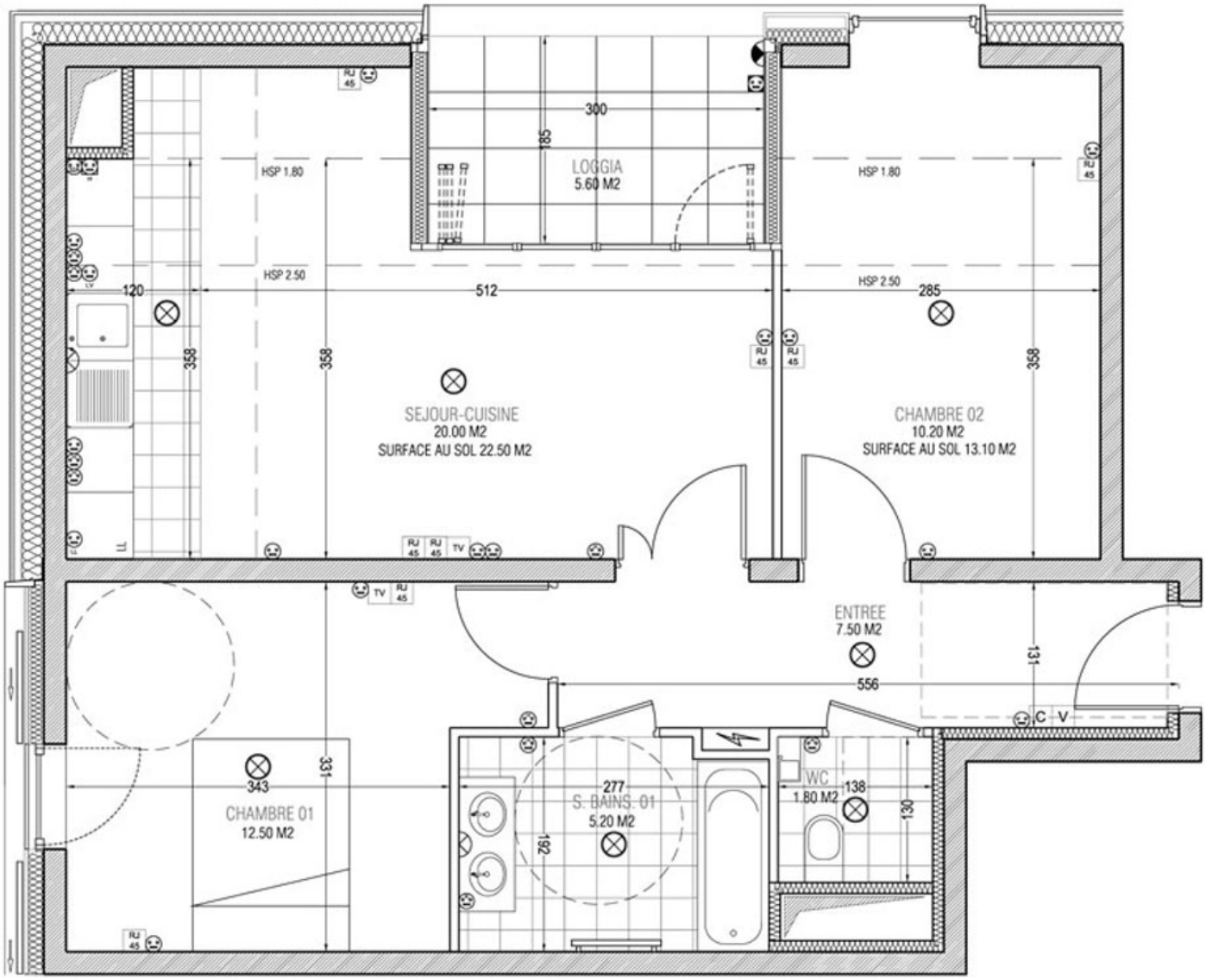
Comme précisé en amont, dans ce mémoire, le programme a été codé en Java par l'intermédiaire de l'environnement de développement (IDE : integrated development environment) Eclipse. Ce choix est essentiellement dû à mes connaissances préliminaires et d'autres auraient très certainement été plus judicieux, mais cela m'a permis de ne pas perdre trop de temps à apprendre la programmation.

Dans la page ci-contre on peut voir l'interface de travail :
- Le rectangle rouge correspond à la fenêtre du programme présentée dans la partie concernant les bases du programme
- Le rectangle orange correspond à l'interface d'Éclipse divisée en plusieurs parties
- Le rectangle vert sert à se déplacer dans les diverses classes composant le programme
- Le rectangle bleu correspond au code (le code affiché ici n'est pas lié à ce qu'affiche la fenêtre du programme, en effet, les fonctionnements d'éclipse et du programme sont indépendants une fois le programme lancé)
- Le rectangle violet correspond à la console, cette console est très utile au développement et permet d'afficher certains messages ou certaines variables lors de l'exécution du programme. Contrairement au code, la console est liée au programme lors de son fonctionnement.


eclipse-workspace - Parametric Design - Beyler/src/Grid.java - Eclipse
Package Explorer
Parametric Design
Parametric Design - Beyler
JComponent.clas
Grid.java
Main.java
cellSelect.typo = TypologieVoid;
cellSelect.bruit1 = TypologieVoid.coefLum2;
cellSelect.lumi1 = TypologieVoid.coefLum;
cellSelect.priorite = TypologieVoid.priorite;
cellSelect.couleur = TypologieVoid.color;
typoSelect.nombreImpose --;
cellListImmuable.remove(cellSelect);
typoSelect.nombreImpose --;
ecran.setText(str);
class comboAction implements ActionListener {
public void actionPerformed(ActionEvent e){
int i = combo.getSelectedIndex();
typoSelect = typoList.get(i);
comboColor.setSelectedIndex(typoSelect.color);
etatSelection = false;
check1.setSelected(false);
refreshTypo();
jtfNomTypo.setText(typoSelect.nom);
if (i == 0 || i == 1) {
check1.setSelected(true);
etatSelection = true;
static public class comboMother implements ActionListener{
public void actionPerformed(ActionEvent e) {
Problems
Javadoc
Declaration
Console
Coverage
Main [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_31.jdk/Contents/H
Writable
Smart Insert
961 : 31
GridBagLayout
Cellule
Typologie
Ok
Rouge
Nombre de typologie(s) :
1
Vali...
Typologie Séléctionnée :
Typologie
Général
Ajouter une cellule
Repartition : 0 %
0
20
40
60
80
100
Priorité : 0
0
20
40
60
80
100
1
Generer
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99
Test - Save
Test - Re...
Recover
50.0
50.0
Generate Site
10
10
Generate Division
5.0
5.0
Generate Cells
Prendre en compte la disposition
Prendre en compte la luminosité
Prendre en compte l'acoustique
Score total :
Score de tyopologie :
Score de luminosité :
Score acoustic :

Expérimentation

Avant de procéder à l'expérimentation, il est important de définir un programme architectural. Pour cet exemple, nous chercherons à obtenir l'esquisse d'un logement dans un site simple. Nous devons d'abord expliciter les différentes typologies. Pour établir ces typologies, nous allons utiliser un exemple de plan existant et utiliser ses statistiques de surface et de répartition. Nous opterons pour un plan d'appartement simple, n'impliquant que peu de typologies afin d'éviter de complexifier le problème et de pouvoir analyser plus simplement.
Le plan choisi provient du site : www.archi-id.net



Ce plan comporte 6 typologies différentes : Entrée, chambre, salle de bains, WC, Sejour / Cuisine et Loggia. Il nous faut à présent déterminer les proportions, on utilisera un tableur répertoriant les surfaces respectives.

Typologies	Surface (m2)	% surface
Entrée	7,5	12%
Chambre	22,7	36%
Salle de bains	5,2	8%
WC	1,8	3%
Sejour / Cuisine	20	32%
Loggia	5,6	9%
Total	62,8	100%

On définira ainsi 8 typologies comme suit :

Typologie 1 : Extérieur
Couleur : Blanc - Répartition : 0%
Typologie de base. La typologie Extérieur représente ce qui n'est pas compris dans le logement. Pour plus tard il pourrait être utile de diversifier cette typologie en plusieurs (jardin, rue, etc.). Cette typologie est à l'origine de la génération de lumière dans les autres typologies.

Typologie 2 : Inconstructible plein
Couleur : Noir - Répartition : 0%
Typologie de base. La typologie Inconstructible plein permet essentiellement de créer les bâtiments existants sur le site, contrairement à l'extérieur il ne sont pas source de lumière.

Typologie 3 : Entrée
Couleur : Vert - Répartition : 12%
La typologie d'entrée sert de lien entre toutes les autres typologies. Elle est liée et étendue et servira de typologie mère à presque toutes les autres. On pourra éventuellement lui attribuer une épaisseur de 2 cellules. En ce qui concerne la luminosité, celle-ci est non nécessaire.

Typologie 4 : Chambres
Couleur : Rouge - Répartition : 36%
La typologie chambre sera composée de deux blocs distincts. On choisira donc des typologies rectangulaires ayant pour mère la typologie d'entrée. Contrairement à la typologie précédente, on favorisera une luminosité modérée.

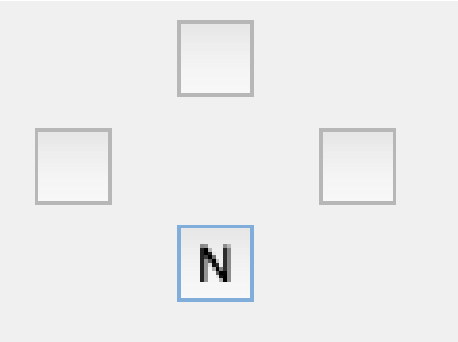
Typologie 5 : Salle de bains
Couleur : Bleu - Répartition : 8%
La typologie Salle de bains sera composée d'un simple bloc. Elle aura comme typologie mère la typologie d'entrée. Afin de la lier à la typologie WC, on définira celle-ci comme typologie fille de la typologie Salle de bains. Il n'est pas nécessaire d'apporter de la lumière naturelle dans cette typologie.

Typologie 6 : WC
Couleur : Violet - Répartition : 3%
Comme la typologie Salle de bains, cette typologie est composée d'un seul bloc non éclairé. On lui attribuera comme typologie mère la typologie Salle de bain.

Typologie 7 : Sejour / Cuisine
Couleur : Orange - Répartition : 32%
La typologie Séjour / Cuisine est la plus étendue, elle sera composée d'un seul bloc condensé et aura comme typologie mère l'Entrée. Cette typologie nécessite un éclairement naturel fort.

Typologie 8 : Loggia
Couleur : Jaune - Répartition : 9%
La typologie Loggia aura comme typologie mère la typologie Sejour / Cuisine. Elle sera composée d'un seul bloc et aura un éclairement naturel fort.

Une fois les 8 typologies définies, il nous reste à définir l'orientation du projet.
Si nous considérons l'emplacement comme étant en région parisienne, nous allons chercher à optimiser l'apport de lumière et de chaleur du soleil, nous allons donc choisir une orientation avec le Nord pointant vers le bas, comme ceci :



Il ne nous reste à présent qu'à définir le nombre d'itérations voulues. Après plusieurs essais, voici un tableau du temps mis par le programme pour générer un nombre donné d'esquisses en fonction du nombre de typologies (répartitions égales). Le temps change certainement en fonction du type de génération choisi pour les typologies, il reste néanmoins un bon indicateur :

Itérations	Nombre de typologies 3	4	5	6	7	8
1	35	36	36	35	41	47
50	797	814	838	869	935	938
100	1 455	1 504	1 423	1 435	1 471	1 622
500	5 074	4 699	5 085	4 851	4 872	5 481
1000	9 611	9 004	8 953	8 593	8 623	9 104
10000	59 092	57 886	60 336	65 395	68 179	73 881

Ces temps ont été obtenus en effectuant la moyenne de 10 essais par résultat. En jaune figurent des temps anormalement élevés par rapport aux autres résultats. Il peut sembler logique qu'un plus grand nombre de typologies entraîne un temps plus long, cependant le nombre de cellules généré est le même étant donné qu'il y a une équirépartition des typologies. Les résultats trop élevés sont sans doute dus d'une part au faible échantillon utilisé pour les calculs, mais d'autre part on remarque qu'ils sont donnés pour un faible nombre de typologies, on peut se demander si l'ordinateur ne sauvegarde pas certaines données en mémoire, ce qui serait représentatif sur un grand nombre d'opérations.
Ces temps sont donc indicatifs et dépendent de l'ordinateur exécutant le programme, voici les caractéristiques de l'ordinateur utilisé pour les calculer :

On peut donc voir qu'il sera possible de générer 10 000 itérations pour notre essai étant donné que cela ne prendra qu'environ 1 minute 15 à l'ordinateur.

Pour effectuer cet essai, nous allons utiliser deux systèmes de sauvegarde présents dans le programme.
Le premier sert à enregistrer les réglages généraux précédemment détaillés et permet ainsi de travailler sur plusieurs cessions, tout en modifiant le code entre-deux. Le système est très rudimentaire et consiste simplement à enregistrer au format texte les typologies dans un fichier à un emplacement précis de l'ordinateur.
Le texte de sauvegarde utilisé pour l'exemple est défini comme suit :

6/50.0/50.0/5.0/5.0/10/10/3/2/12/Entree/5/1/false/0/true/true/2/false/false/2/new/3/36/Chambre/2/18/false/2/false/false/1/false/true/2/new/4/8/SalleDeBains/6/1/false/0/false/false/1/true/true/2/new/5/3/WC/7/1/false/0/false/false/1/true/true/4/new/6/32/SejourCuisine/3/1/false/3/false/false/1/true/true/5/new/7/9/Loggia/4/1/false/3/false/false/1/true/true/2/new/

Regardons de plus près la première ligne et plus précisément ce qui suit :
6/50.0/50.0/5.0/5.0/10/10/3/2/12/Entree/5/1/false/0/true/true/2/false/false/2/new/

Cette séquence correspond à la fois à certaines données du projet (le début) et à une typologie. Vous trouverez page suivante un tableau détaillant les différentes variables. Celles-ci sont séparées par un «/» afin de pouvoir facilement les identifier.

6	Nombre de typologies en plus des typologies de base
50.0/50.0/5.0/5.0/10/10	Données relatives à la grille (dimensions générales, nombre de ligne et de colonne, dimensions d'une cellule)
3	Orientation du projet. Ce chiffre varie entre 1 et 4, ici l'orientation est 3 pour Sud (1: Nord/ 2: Ouest/ 3: Sud / 4: Est)
2/12/Entree/5/1/false/0/true/true/2/false/false/2	Première typologie. En premier on retrouve son numéro d'identification. La typologie Extérieur a le numéro d'identification 0, l'Inconstructible Plein le numéro 1 et ensuite chaque typologie à le sien incrémenté de 1. Viennent ensuite son pourcentage de répartition, son nom, le numéro correspondant à sa couleur et toutes les variables permettant de déterminer sa disposition et son éclairement
/new/	Chaque typologie est séparée par un /new/ afin de facilement lire le texte de sauvegarde si besoin

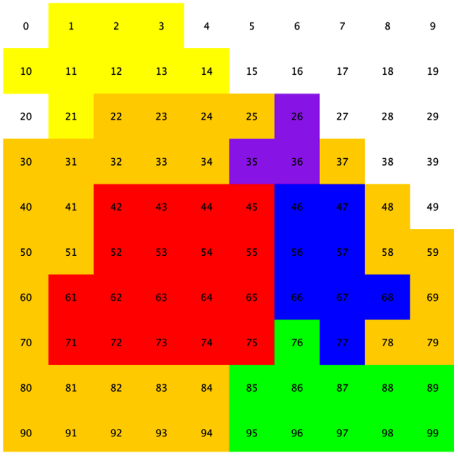
Le second système de sauvegarde sert à conserver la répartition des cellules en mémoire pour ensuite pouvoir les comparer entre elles et revenir sur les itérations voulues. Ce système est lui aussi basé sur un texte décrivant chaque cellule une par une.
Voici un extrait correspondant à la première cellule :
0/100/0/2.5/2.5/0.0/0/false/false/true/0/30000/0/0/30000.0/0/0/0/0/0/0/0/new/

0	Numéro de l'itération (ici première itération : 0)
100	Nombre de cellules composant la grille
0	Numéro de la cellule à laquelle correspondent les données
2.5/2.5/0.0/0	Coordonnées x, y et z de la cellule
false/false/true/0/30000/0/0/30000.0/0/0/0/0/0/0/0	Variables de la cellule, sa valeur de luminosité, de bruit, est-ce qu'elle transmet la lumière, etc.
/new/	Encore une fois les cellules sont séparées par un /new/ afin de lire facilement le texte de sauvegarde.

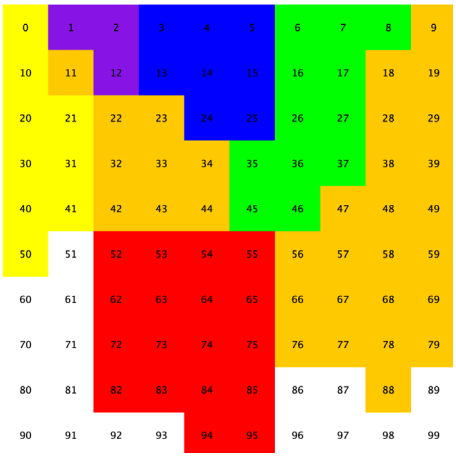
La description de ces deux systèmes de sauvegarde permet de mieux conceptualiser le fonctionnement du programme. Il existe de bien meilleurs moyens de garder en mémoire des données, il serait facile de conserver directement les listes de cellule et de typologies, l'avantage ici était de pouvoir conserver ces valeurs même en fermant le programme et de pouvoir les manipuler à guise. Encore une fois, si l'on garde en vue le développement d'une arborescence présentant un large panel d'esquisses et permettant de revenir au stade de génération voulu, il faut envisager un système de base de données beaucoup plus complexe.

2) Generation

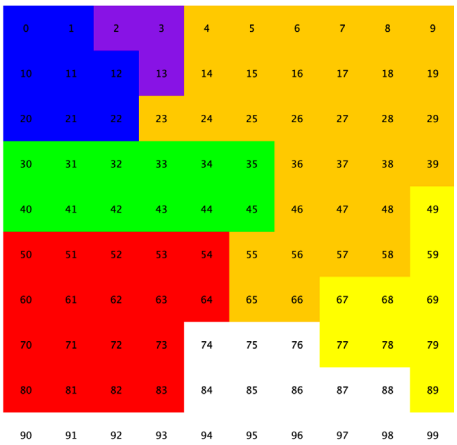
Nous allons à présent étudier la génération d'une itération d'esquisse de plan. Étant donné que le programme n'est pas parfait, il est encore difficile de s'appuyer sur les scores pour les départager (on étudiera cela plus en détail dans la partie suivante), j'ai donc opté pour la génération de 100 plans afin de pouvoir les évaluer par moi-même.
Voici quelques plans ainsi que leurs scores respectifs (par ordre croissant de score) :



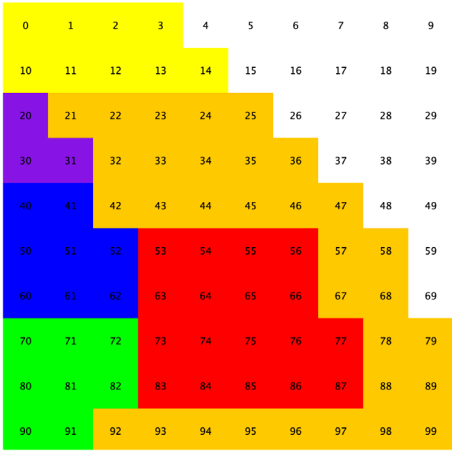
Iteration 65 - Score 58%



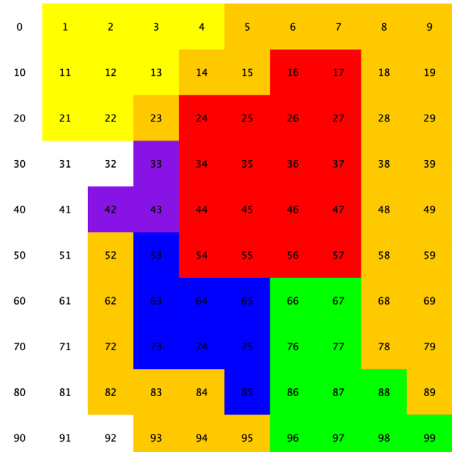
Iteration 83 - Score 54%



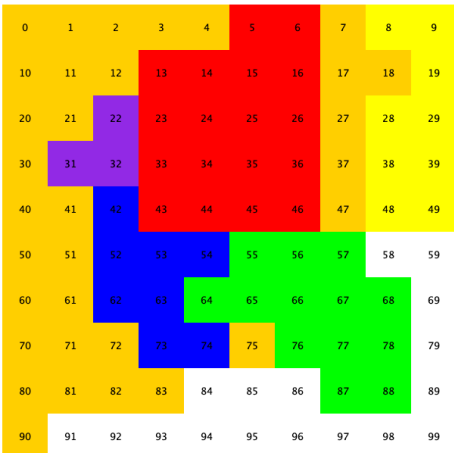
Iteration 25 - Score 51%



Iteration 69 - Score 49%



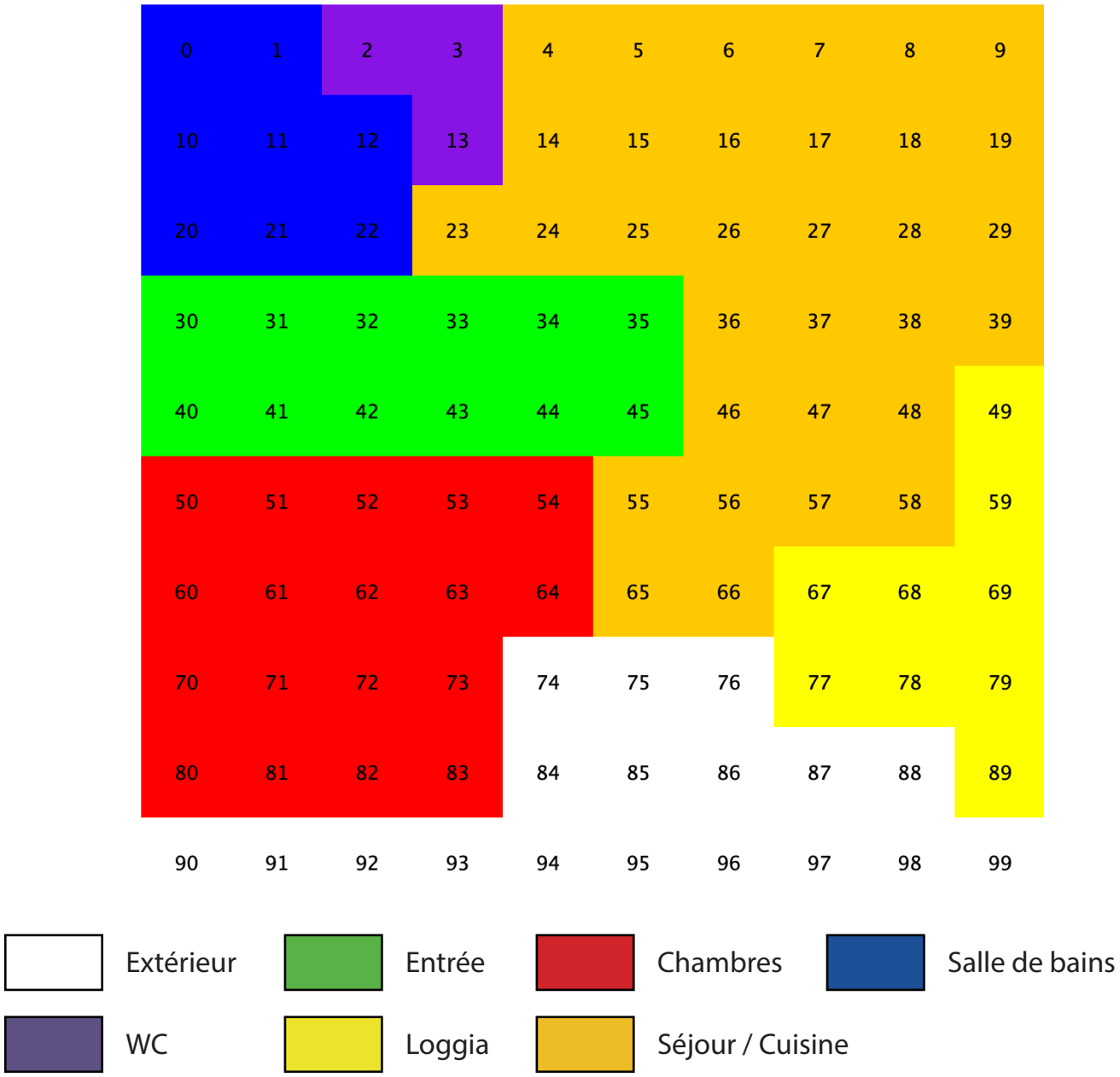
Iteration 75 - Score 40%



Iteration 13 - Score 32%

Comme explicité précédemment, les scores ne sont pas entièrement représentatifs, en effet, il faudrait coefficienter certaines parties, nous détaillerons cela plus tard.

Pour la suite, nous allons étudier l'Itération 25, qui me semble être la plus pertinente parmi les différentes itérations présentées.



Cette itération a le mérite de présenter une répartition relativement homogène. Elle est orientée Nord, ce qui baisse énormément son score de luminosité, cependant son score typologique remonte beaucoup son score total grâce à son coefficient de variation très faible.

Afin de pouvoir regarder en détail la génération de cette disposition, j'ai demandé au programme d'expliciter certaines variables au fur et à mesure, c'est ce qu'on appelle les Logs (Log File) ou l'Historique en français. Cet Historique apparaît dans la console d'Eclipse.
Voici, ci-contre, les logs concernant la génération :

Debut du recover de projet

6/50.0/50.0/5.0/5.0/10/10/3/2/12/Entree/5/1/false/0/true/true/2/false/false/0/new/3/36/Chambre/2/18/false/2/false/false/1/false/true/2/new/4/8/SalleDeBains/6/1/false/0/false/false/1/true/true/2/new/5/3/WC/7/1/false/0/false/false/1/true/true/4/new/6/32/SejourCuisine/3/1/false/3/false/false/1/true/true/2/new/7/9/Loggia/4/1/false/3/false/false/1/true/true/6/new/

Ajout de la typo : Entree
Ajout de la typo : Chambre
Ajout de la typo : SalleDeBains
Ajout de la typo : WC
Ajout de la typo : SejourCuisine
Ajout de la typo : Loggia

Attribution Lancée : Extérieur
Case typo = 1 / Mother = false / Liee = false / Condensee = false
Iteration simple lancée

Attribution Lancée : Inconstructible plein
Case typo = 1 / Mother = false / Liee = false / Condensee = false
Iteration simple lancée

Attribution Lancée : Entree
Case typo = 1 / Mother = false / Liee = true / Condensee = false

Ajout de la cellule 35
Ajout de la cellule 34
Ajout de la cellule 33
Ajout de la cellule 32
Ajout de la cellule 31
Ajout de la cellule 30
Ajout de la cellule 45
Ajout de la cellule 44
Ajout de la cellule 43
Ajout de la cellule 42
Ajout de la cellule 41
Ajout de la cellule 40

Attribution Lancée : Chambre
Case typo = 18 / Mother = true / Liee = false / Condensee = false

Ajout de la cellule 50
Ajout de la cellule 60
Ajout de la cellule 70
Ajout de la cellule 80
Ajout de la cellule 51
Ajout de la cellule 61
Ajout de la cellule 71
Ajout de la cellule 81
Ajout de la cellule 52
Ajout de la cellule 62
Ajout de la cellule 72
Ajout de la cellule 82
Ajout de la cellule 53
Ajout de la cellule 63
Ajout de la cellule 73
Ajout de la cellule 83
Ajout de la cellule 54
Ajout de la cellule 64

Attribution Lancée : SalleDeBains
Case typo = 1 / Mother = true / Liee = false / Condensee = true

Ajout de la cellule 20
Ajout de la cellule 10
Ajout de la cellule 21
Ajout de la cellule 11
Ajout de la cellule 0
Ajout de la cellule 22
Ajout de la cellule 1
Ajout de la cellule 12

Attribution Lancée : WC
Case typo = 1 / Mother = true / Liee = false / Condensee = true

Ajout de la cellule 2
Ajout de la cellule 3
Ajout de la cellule 13

Attribution Lancée : SejourCuisine
Case typo = 1 / Mother = true / Liee = false / Condensee = true

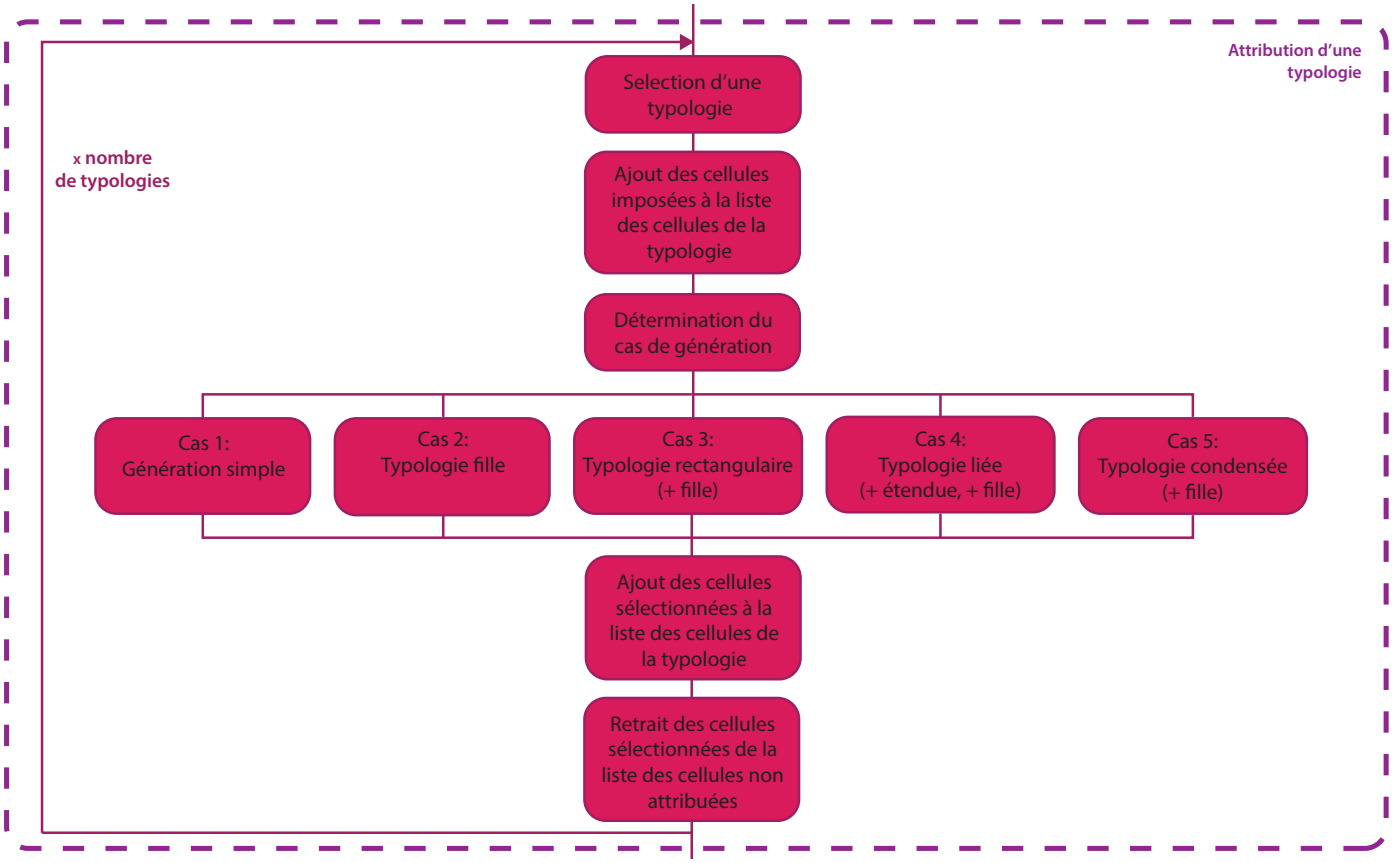
Ajout de la cellule 14
Ajout de la cellule 4
Ajout de la cellule 15
Ajout de la cellule 24
Ajout de la cellule 5
Ajout de la cellule 23
Ajout de la cellule 25
Ajout de la cellule 16
Ajout de la cellule 6
Ajout de la cellule 26
Ajout de la cellule 36
Ajout de la cellule 17
Ajout de la cellule 7
Ajout de la cellule 27
Ajout de la cellule 37
Ajout de la cellule 46
Ajout de la cellule 18
Ajout de la cellule 8
Ajout de la cellule 28
Ajout de la cellule 55
Ajout de la cellule 47
Ajout de la cellule 38
Ajout de la cellule 56
Ajout de la cellule 19
Ajout de la cellule 48
Ajout de la cellule 57
Ajout de la cellule 9
Ajout de la cellule 29
Ajout de la cellule 65
Ajout de la cellule 39
Ajout de la cellule 66
Ajout de la cellule 58

Attribution Lancée : Loggia
Case typo = 1 / Mother = true / Liee = false / Condensee = true

Ajout de la cellule 59
Ajout de la cellule 49
Ajout de la cellule 69
Ajout de la cellule 68
Ajout de la cellule 79
Ajout de la cellule 67
Ajout de la cellule 78
Ajout de la cellule 77
Ajout de la cellule 89

Comme expliqué dans la partie précédente, les deux premiers paragraphes correspondent au système de sauvegarde et plus précisément à la récupération des différentes données.

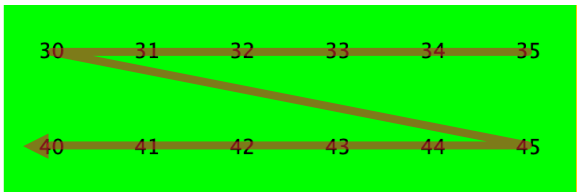
Le programme va ensuite générer la grille typologie par typologie en fonction des différents paramètres.



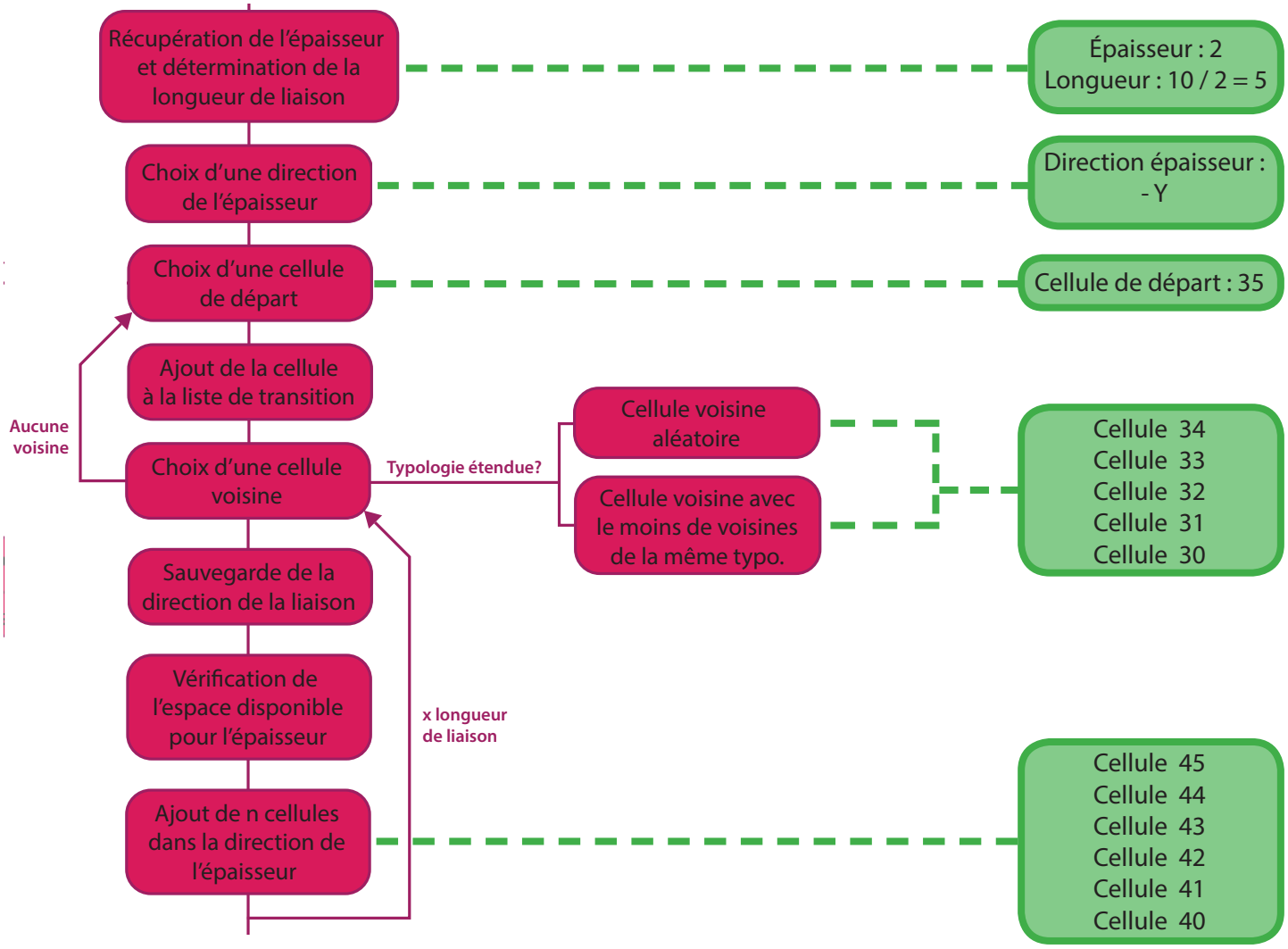
Les deux premières typologies ne sont pas intéressantes dans notre exemple étant donné que nous n'avons pas défini un pourcentage de répartition pour celles-ci.

La typologie d'entrée est, en revanche, beaucoup plus utile. Il s'agit de la typologie de base du programme, elle permet de relier entre elles les autres typologies. Le programme donne :
Case typo = 1 / Mother = false / Liee = true / Condensee = false

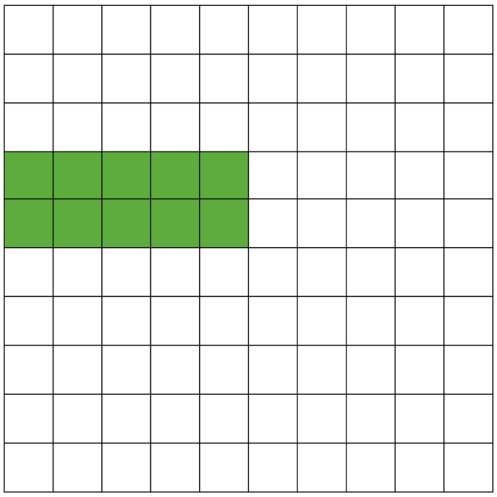
Autrement dit, il s'agit d'une typologie liée, pour rappel, nous avons demandé à ce qu'elle soit liée, étendue et d'épaisseur deux, voici donc comment le programme l'a généré :
- Il a choisi une cellule au hasard (**35**) puis a choisi une direction aléatoire (**34**)
- Comme la typologie est étendue, le programme a cherché à tourner le moins possible, ce qui était possible étant donnée que la grille était alors libre
- Il a ensuite doublé son épaisseur en repartant de la cellule (**45**) jusqu'à la cellule (**40**)



Extrait de description de l'algorithme d'attribution d'une typologie liée, étendue possédant une épaisseur :

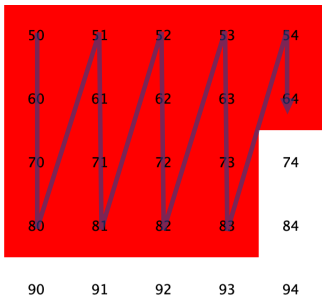


État actuel de la grille :

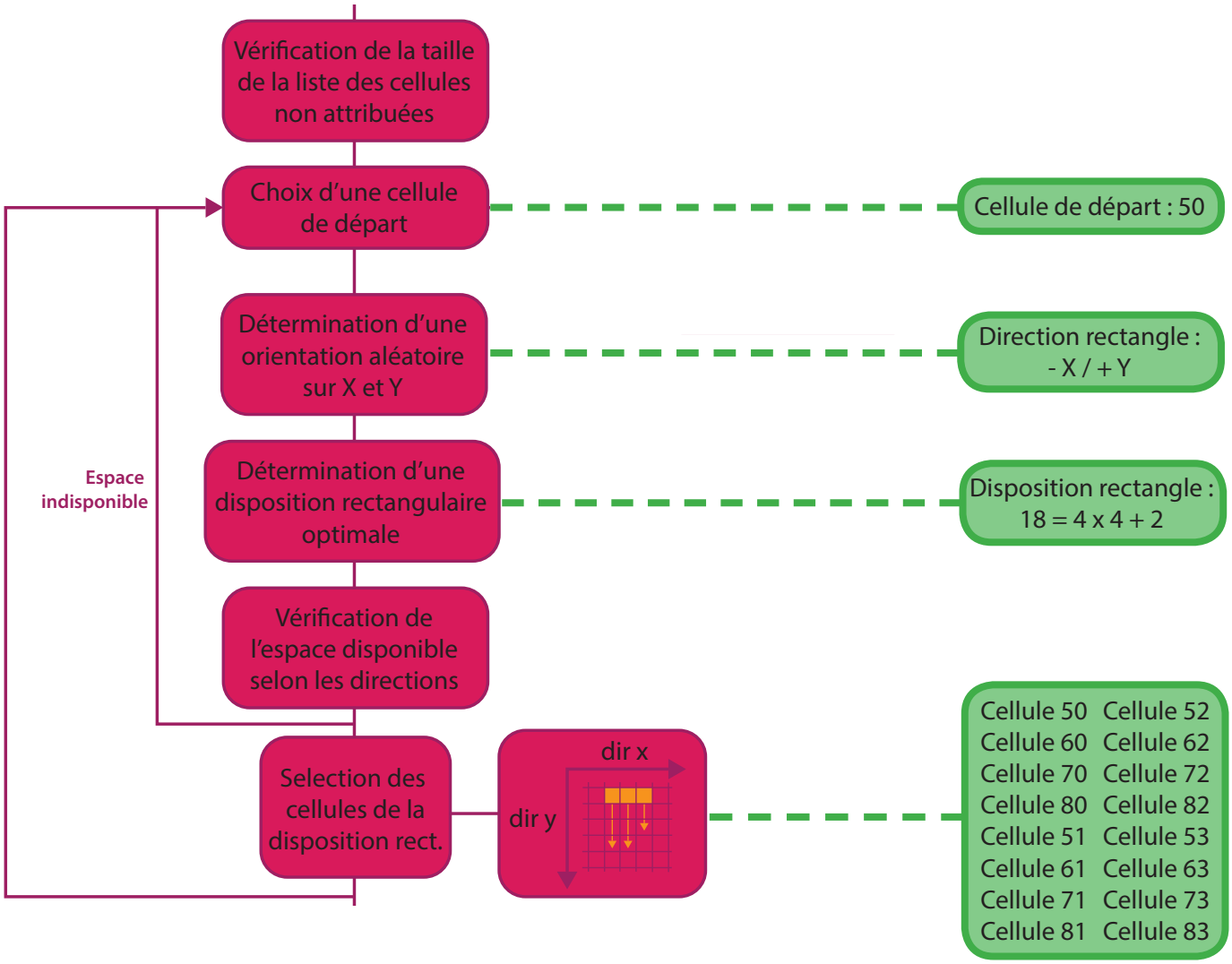


La typologie suivante est la typologie de la chambre. Le programme la définit comme suit :
Case typo = 18 / Mother = true / Liee = false / Condensee = false

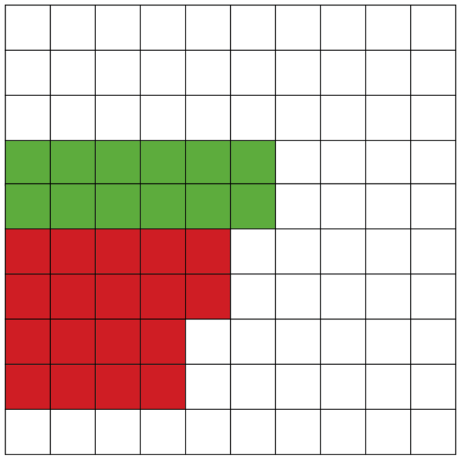
C'est donc une typologie rectangulaire de 18 cellules ayant pour mère la typologie Entrée. Voici comment le programme l'a générée :
- Il a choisi une cellule aléatoire adjacente à la typologie Entrée (**50**)
- Il a ensuite défini une direction sur X et Y puis une répartition rectangulaire optimale pour 18 cellules (**4x4 + 2**)
- Il a enfin vérifié si l'espace disponible était suffisant et a réparti comme suit :



Extrait de description de l'algorithme d'attribution d'une typologie rectangulaire :

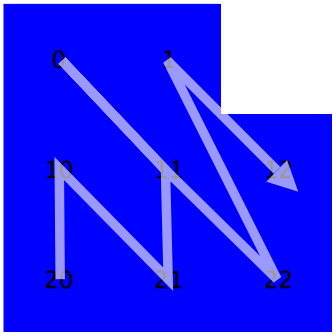


État actuel de la grille :

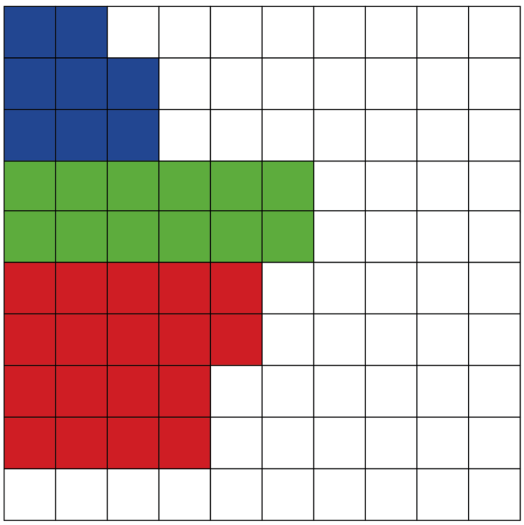


Le programme a ensuite généré la typologie Salle de bains en la définissant comme ceci :
Case typo = 1 / Mother = true / Liee = false / Condensee = true

Il s'agit d'une typologie condensée ayant comme mère la typologie Entrée. Le programme la génère ainsi :
- Il a choisi une cellule aléatoire adjacente à la typologie Entrée (**20**)
- Il tri les cellules disponibles par proximité avec la cellule (**20**) pour rappel, les cellules en dessous de la cellule (**20**) sont occupées par la typologie Entrée
- Il sélectionne ensuite les 7 plus proches cellules par ordre de proximité.



État actuel de la grille :

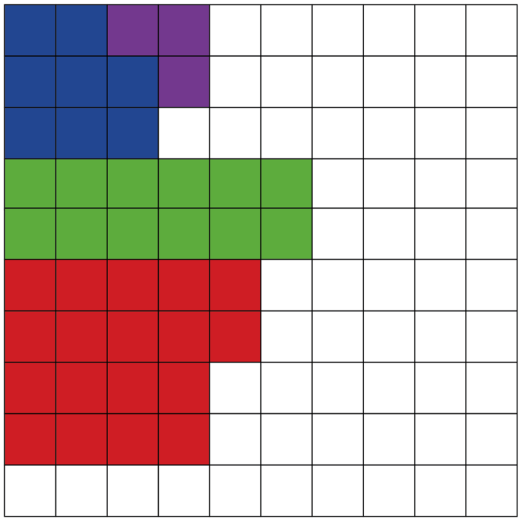


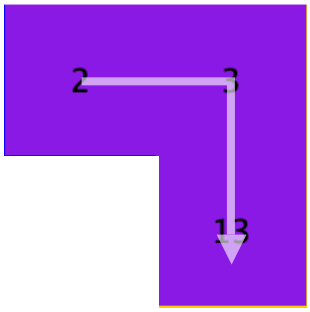

Vérification de la taille de la liste des cellules non attribuées
Choix d'une cellule de départ
Cellule de départ : 20
Création de la liste des cellules ordonnées par leur distance par rapport à la cellule de départ
Sélection de la liste des cellules libres
Cellules 0 à 9
Cellules 10 à 19
Cellules 21 à 29
Cellules 36 à 39
Cellules 46 à 49
Cellules 55 à 59
Cellules 65 à 69
Cellules 64 à 79
Cellules 84 à 89
Cellules 90 à 99
Sélection d'une cellule parmi la liste
Exemple Cellule 0
Calcul de la distance de la cellule par rapport à la cellule de départ
Calcul de distance (0) :
Dx = X20 - X0 = 2.5 - 2.5 = 0
Dy = Y20 - Y0 = 12.5 - 2.5 = 10
D = √(Dx² - Dy²) = 10
Modification de l'attribut « distance » de la cellule
Distance Cellule 0 = 10
Tri de la liste en fonction des attributs « distance » des cellules
Cellule 10
Cellule 21
Cellule 11
Cellule 0
Cellule 22
Cellule 1
Cellule 12
Cellule 2
Cellule 23
Cellule 13
Cellule 3
Cellule 24
Cellule 14
Cellule 4
Cellule 25
Cellule 15
Cellule 5
Cellule 26
Sélection des n premières cellules
n = 7

Suite à la typologie Salle de bains, la typologie WC a été générée de la façon suivante :
Case typo = 1 / Mother = true / Liee = false / Condensee = true

Il s'agit là encore d'une typologie condensée ayant cette fois pour mère la typologie Salle de bains. Le raisonnement est donc similaire à la typologie précédente.

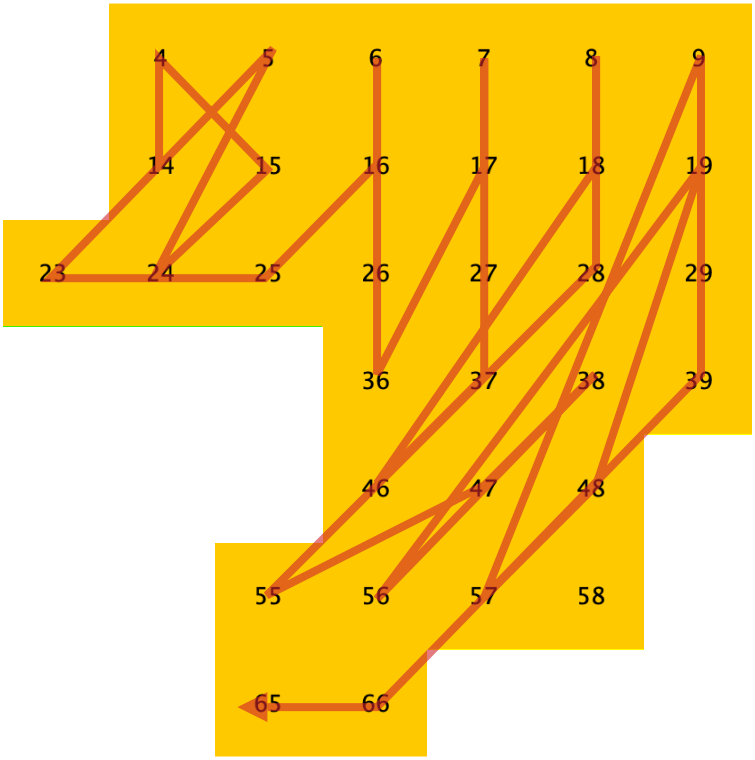
État actuel de la grille :



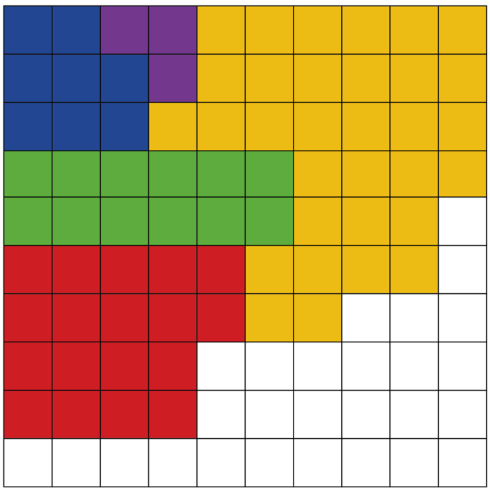


La typologie suivante est la typologie du séjour et de la cuisine. Elle est définie comme suit :
Case typo = 1 / Mother = true / Liee = false / Condensee = true

C'est donc encore une fois une typologie condensée ayant pour mère la typologie Entrée. La génération est donc semblable aux deux précédentes.

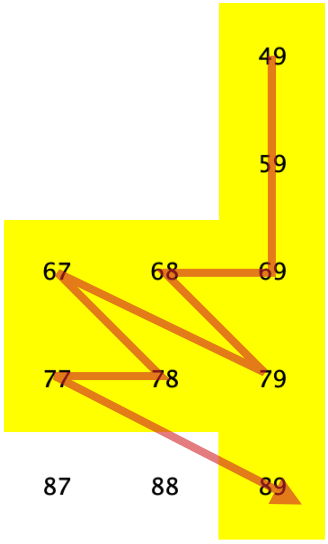


État actuel de la grille :

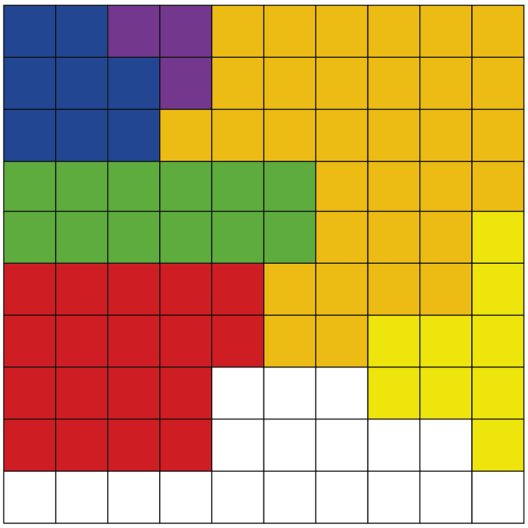


La dernière typologie est la typologie Loggia définie comme suit :
Case typo = 1 / Mother = true / Liee = false / Condensee = true

À l'instar des typologies précédentes, elle est condensée et a une typologie mère, qui est cette fois la typologie séjour / cuisine.



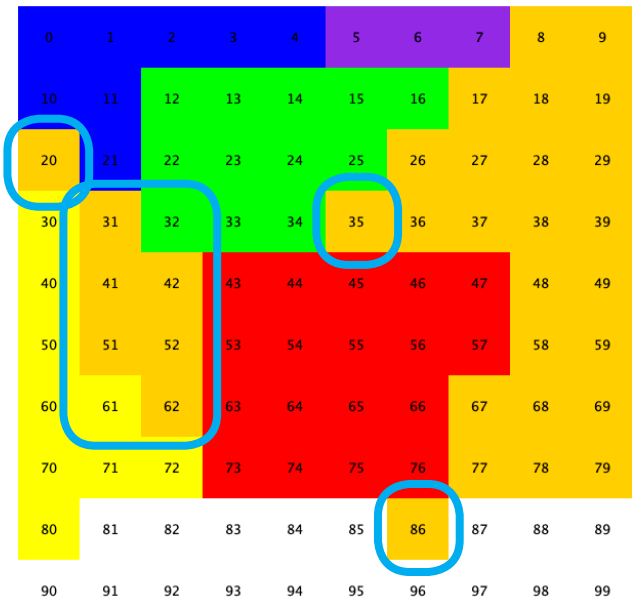
État final de la grille :



Voici donc comment cette itération a été générée. On peut cependant relever quelques erreurs dues à la façon de générer cette grille que l'on peut retrouver dans les autres itérations.

Problème 1 : Définition de la typologie condensée

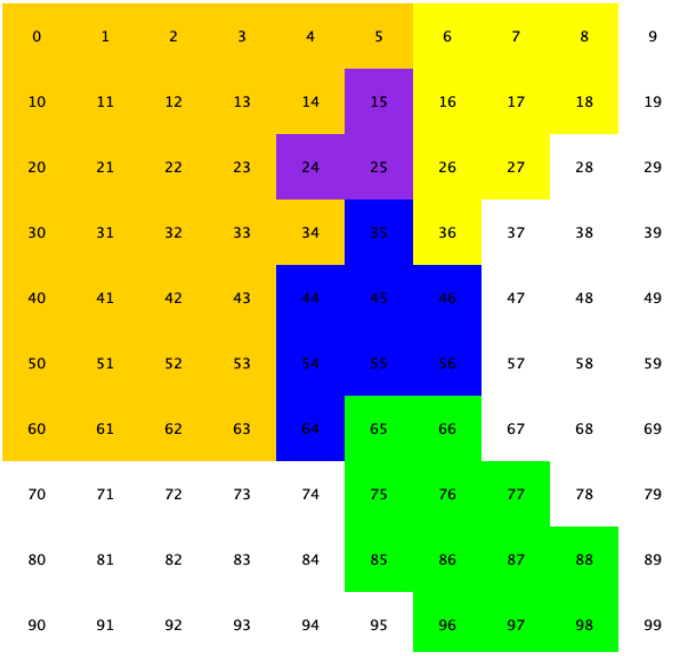
La façon dont est définie la typologie condensée se base exclusivement sur la distance entre les cellules, ce qui peut engendrer des répartitions non voulues.



Si l'on prend l'itération ci-contre en exemple, et que l'on considère la cellule (**35**) comme cellule de départ, alors les cellules (**20**), (**85**) ainsi que le groupe de cellules (**31-41-42-52-52-62**) sont plus proches de la cellule (**35**) que la cellule (**79**) en termes de distance. Cependant celles-ci sont isolées du reste. En théorie cette répartition est pénalisée lors de l'attribution des scores, il n'empêche qu'elle peut se produire. Pour parer cela, il serait possible de tout d'abord créer la liste des cellules non isolées de la cellule de départ puis de trier cette liste par distance et d'attribuer la typologie par ordre de proximité au sein de cette liste.

Problème 2 : Attribution de la typologie rectangulaire

Lorsque l'on demande au programme d'attribuer une typologie rectangulaire d'un certain nombre de cellules de base, il va tout d'abord choisir une direction aléatoire sur X et Y, puis voir s'il y a suffisamment de place pour attribuer les cellules, s'il n'y a pas d'espace suffisant, il va alors chercher d'autres directions d'orientation. Si aucune ne fonctionne, il va simplement passer à la typologie suivante et ne pas générer cette typologie, du moins pas entièrement.



Si l'on prend l'itération ci-contre en exemple, il n'y a pas de cellules attribuées à la typologie chambre étant donné que le programme n'a pas trouvé d'espace suffisant. Ne pas générer de typologie si celle-ci ne peut pas être mise est essentiel au programme, cela permet de ne pas tourner indéfiniment en boucle, il serait cependant intéressant de trouve rune alternative.

Une solution serait d'exclure de telles itérations lors du calcul du score. En effet, pour l'instant un score typologique est pris en compte, mais il ne concerne que la recherche de créer des tendances, pas si les consignes de départ sont bien respectées (existe-t-il le bon nombre de cellules par typologie, etc. ?).

Une seconde solution serait de créer une attitude de génération alternative comme par exemple imposer une typologie rectangulaire avec moins de cellules de base ou de changer en typologie condensée.

Problème 3 : Typologies orphelines & mères inégales

Les typologies mères sont à l'origine de nombreux problèmes. Le plus flagrant est celui des typologies orphelines. À l'image des typologies rectangulaires qui ne se génèrent pas, si une typologie est fille d'une autre typologie, mais que celle-ci n'a pas d'espace libre à ses côtés, alors la typologie «orpheline» ne sera pas générée. Cela arrive avec un grand nombre de typologies. De même, lorsqu'une typologie mère a plusieurs typologies filles, celles-ci sont générées à la suite et non simultanément, ce qui fait que les dernières sont beaucoup moins bien gérées que les premières, voir peuvent disparaître.

Pour parer à ce problème, une des solutions serait d'attribuer de façon équitable les cellules mères aux typologies filles en imposant dès la création de la typologie mère une cellule adjacente fille de chaque typologie, on repartirait ensuite de ces cellules pour générer les typologies filles. Une seconde solution serait de définir des typologies mères de substitution pour certains cas bien précis de saturation des typologies mères d'origine.

Encore une fois, ces cas peuvent être gérés via le calcul des scores en vérifiant le respect des paramètres de base, il reste néanmoins intéressant de trouver des substituts dès la génération afin d'éviter la perte d'une itération.

3) Résultats

Après avoir étudié le fonctionnement d'une itération dans un cas précis, nous allons voir comment le programme attribue un score à cette itération afin de pouvoir la comparer. Le score permet d'affiner les résultats en favorisant les plans répondant aux critères souhaités, en effet, comme vu dans la partie précédente, la génération est loin d'être parfaite et de suffire en elle-même. Cependant, le calcul actuel du score est encore incomplet et mériterait de nombreux ajustements, il permet néanmoins de déceler certains résultats et il est important de comprendre ses différentes failles afin de pouvoir les corriger ultérieurement.
Voici ci-dessous, les logs du calcul des scores :

Calcul du score de luminosité

Calcul du taux de luminosité de chaque cellule

Cellule 0 - Luminosité : 0.0 / Cellule 1 - Luminosité : 0.0 / Cellule 2 - Luminosité : 0.0 / Cellule 3 - Luminosité : 0.0 / Cellule 4 - Luminosité : 0.0 / Cellule 5 - Luminosité : 0.0 / Cellule 6 - Luminosité : 0.0 / Cellule 7 - Luminosité : 0.0 / Cellule 8 - Luminosité : 0.0 / Cellule 9 - Luminosité : 0.0 / Cellule 10 - Luminosité : 0.0 / Cellule 11 - Luminosité : 0.0 / Cellule 12 - Luminosité : 0.0 / Cellule 13 - Luminosité : 0.0 / Cellule 14 - Luminosité : 0.0 / Cellule 15 - Luminosité : 0.0 / Cellule 16 - Luminosité : 0.0 / Cellule 17 - Luminosité : 0.0 / Cellule 18 - Luminosité : 0.0 / Cellule 19 - Luminosité : 0.0 / Cellule 20 - Luminosité : 0.0 / Cellule 21 - Luminosité : 0.0 / Cellule 22 - Luminosité : 0.0 / Cellule 23 - Luminosité : 0.0 / Cellule 24 - Luminosité : 0.0 / Cellule 25 - Luminosité : 0.0 / Cellule 26 - Luminosité : 0.0 / Cellule 27 - Luminosité : 0.0 / Cellule 28 - Luminosité : 0.0 / Cellule 29 - Luminosité : 0.0 / Cellule 30 - Luminosité : 0.0 / Cellule 31 - Luminosité : 0.0 / Cellule 32 - Luminosité : 0.0 / Cellule 33 - Luminosité : 0.0 / Cellule 34 - Luminosité : 0.0 / Cellule 35 - Luminosité : 0.0 / Cellule 36 - Luminosité : 0.0 / Cellule 37 - Luminosité : 0.0 / Cellule 38 - Luminosité : 0.0 / Cellule 39 - Luminosité : 0.0 / Cellule 40 - Luminosité : 0.0 / Cellule 41 - Luminosité : 0.0 / Cellule 42 - Luminosité : 0.0 / Cellule 43 - Luminosité : 0.0 / Cellule 44 - Luminosité : 0.0 / Cellule 45 - Luminosité : 0.0 / Cellule 46 - Luminosité : 0.0 / Cellule 47 - Luminosité : 0.0 / Cellule 48 - Luminosité : 0.0 / Cellule 49 - Luminosité : 0.0 / Cellule 50 - Luminosité : 0.0 / Cellule 51 - Luminosité : 0.0 / Cellule 52 - Luminosité : 0.0 / Cellule 53 - Luminosité : 0.0 / Cellule 54 - Luminosité : 48.0 / Cellule 55 - Luminosité : 48.0 / Cellule 56 - Luminosité : 49.92 / Cellule 57 - Luminosité : 0.0 / Cellule 58 - Luminosité : 0.0 / Cellule 59 - Luminosité : 0.0 / Cellule 60 - Luminosité : 0.0 / Cellule 61 - Luminosité : 0.0 / Cellule 62 - Luminosité : 0.0 / Cellule 63 - Luminosité : 82.08 / Cellule 64 - Luminosité : 1200.0 / Cellule 65 - Luminosité : 1201.92 / Cellule 66 - Luminosité : 1248.0 / Cellule 67 - Luminosité : 84.0 / Cellule 68 - Luminosité : 52.665600000000005 / Cellule 69 - Luminosité : 4.1289830400000005 / Cellule 70 - Luminosité : 48.0 / Cellule 71 - Luminosité : 49.92 / Cellule 72 - Luminosité : 84.55680000000001 / Cellule 73 - Luminosité : 951.3822720000001 / Cellule 74 - Luminosité : 30000.0 / Cellule 75 - Luminosité : 30000.0 / Cellule 76 - Luminosité : 30000.0 / Cellule 77 - Luminosité : 2100.0 / Cellule 78 - Luminosité : 1316.64 / Cellule 79 - Luminosité : 183.26075228160002 / Cellule 80 - Luminosité : 1201.92 / Cellule 81 - Luminosité : 1248.0768 / Cellule 82 - Luminosité : 1287.305472 / Cellule 83 - Luminosité : 2184.5568000000003 / Cellule 84 - Luminosité : 30000.0 / Cellule 85 - Luminosité : 30000.0 / Cellule 86 - Luminosité : 30000.0 / Cellule 87 - Luminosité : 30000.0 / Cellule 88 - Luminosité : 30000.0 / Cellule 89 - Luminosité : 2104.12898304 / Cellule 90 - Luminosité : 30000.0 / Cellule 91 - Luminosité : 30000.0 / Cellule 92 - Luminosité : 30000.0 / Cellule 93 - Luminosité : 30000.0 / Cellule 94 - Luminosité : 30000.0

Cellule 95 - Luminosité : 30000.0 / Cellule 96 - Luminosité : 30000.0 / Cellule 97 - Luminosité : 30000.0 / Cellule 98 - Luminosité : 30000.0 / Cellule 99 - Luminosité : 30000.0 /

Calcul du nombre de cellules respectant l'intervalle de luminosité de leur typologie

Typologie étudiée : Entree / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : Chambre / Intervalle : 600.0 - 1200.0 Lux
Score de luminosité de la typologie : 5.0 %

Typologie étudiée : SalleDeBains / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : WC / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : SejourCuisine / Intervalle : 1200.0 - 30000.0 Lux
Score de luminosité de la typologie : 6.0 %

Typologie étudiée : Loggia / Intervalle : 1200.0 - 30000.0 Lux
Score de luminosité de la typologie : 33.0 %

Score de luminosité total : 7.33 %

Calcul du score de la typologie Entree
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 20.203050891044214
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

Calcul du score de la typologie Chambre
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 35.355339059327385
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

Calcul du score de la typologie SalleDeBains
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 29.397236789606563
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

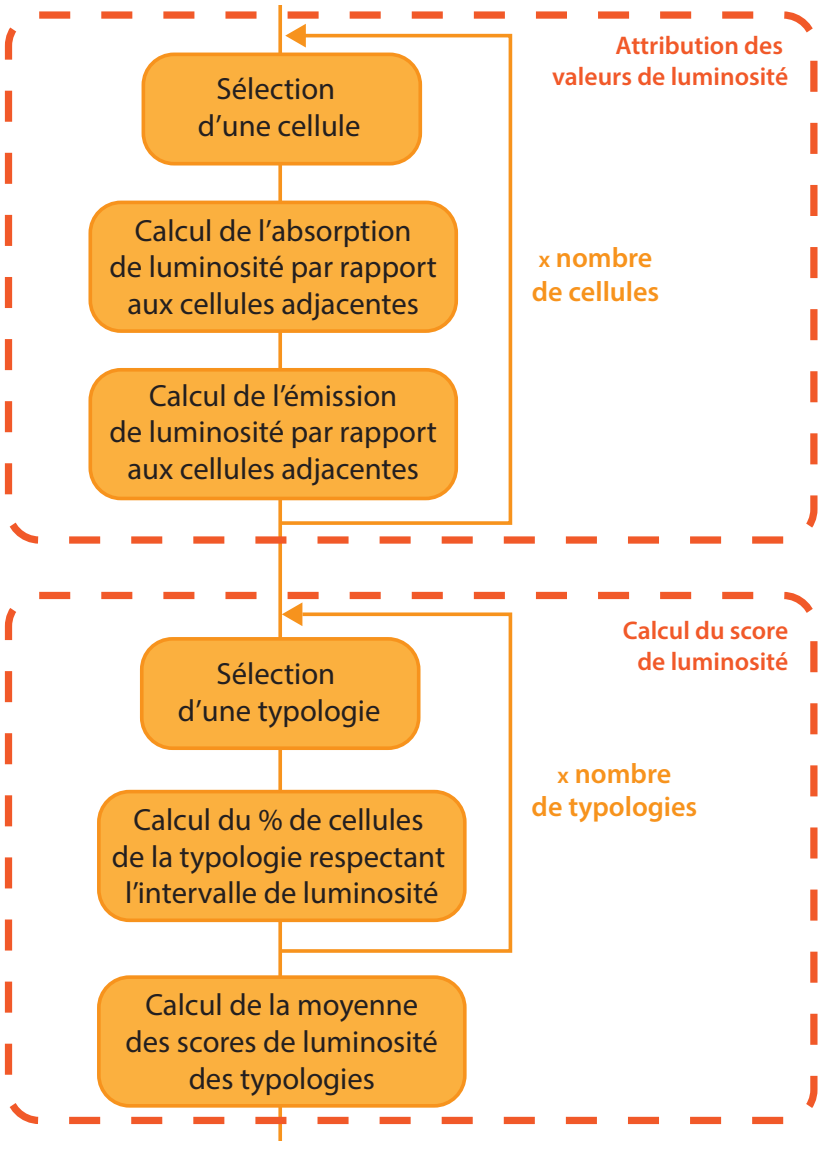
Calcul du score de la typologie WC
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 0.0
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

Calcul du score de la typologie SejourCuisine
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 46.53266869843537
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

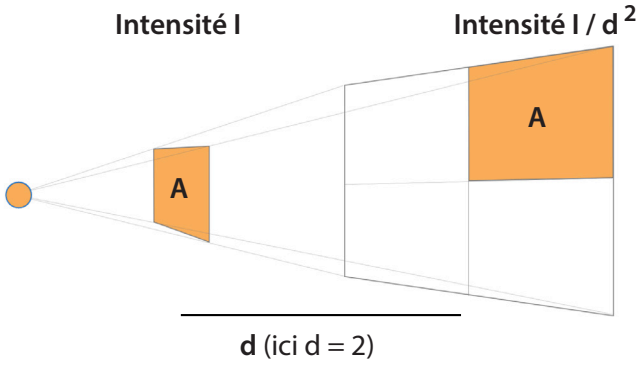
Calcul du score de la typologie Loggia
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 18.708286933869704
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

Coefficient de variation moyen du nombre de typologies voisines : 25.032763728713874
Coefficient de variation moyen du nombre de groupe : 0.0
Coefficient de variation moyen de distance entre les groupes : NaN
Coefficient de variation moyen de distance interne des groupes : 0.0
Coefficient de variation total : 6.258190932178469
Score typologique total : 93.74 %

Le programme va ainsi, dans un premier temps, calculer le score de luminosité puis, dans un second temps, calculer le score typologique de l'itération. Ce sont, pour le moment, les deux seuls critères implantés dans l'algorithme, bien d'autres peuvent être envisagés pour la suite.
Voici pour rappel un extrait de description de l'algorithme du calcul de luminosité :



Le programme va tout d'abord calculer le taux de luminosité en Lux de chaque cellule. Ce score est basé sur le modèle physique décrit dans la partie *Critères et modélisation* prenant en compte la distance entre les cellules, celle-ci au carré correspondant à la perte de puissance en Lux.



Ce modèle permet de connaître la quantité de luminosité absorbée par chaque cellule ainsi que la quantité émise. Certaines typologies vont permettre le passage de la lumière, c'est généralement le cas des circulations ou encore des espaces extérieurs tandis que d'autres auront un comportement opaque. C'est également ici que l'orientation du site entre en jeu, la lumière naturelle provenant du Nord est réduite à 50% de son intensité initiale, tandis que la lumière naturelle provenant de l'Ouest ou de l'Est est réduite à 75% de son intensité initiale. Ces coefficients sont assez arbitraires et ne correspondent pas à une valeur physique, ils permettent néanmoins de favoriser les orientations des typologies en fonction de nos choix architecturaux.

Une fois que chaque cellule possède sa valeur de luminosité, le programme va chercher à voir si cette valeur correspond au critère demandé à la typologie. Pour ce faire, l'algorithme consiste à étudier l'intervalle de luminosité dans lequel se trouve la cellule. Pour rappel, les intervalles sont définis comme suit :
Éclairement naturel fort : 1200 - 30000 Lux
Éclairement naturel modéré : 600 - 1200 Lux
Éclairement naturel faible : 50 - 600 Lux
Éclairement non nécessaire : 0 - 50 Lux

Voici ce que donnent les logs :

Typologie étudiée : Entree / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : Chambre / Intervalle : 600.0 - 1200.0 Lux
Score de luminosité de la typologie : 5.0 %

Typologie étudiée : SalleDeBains / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : WC / Intervalle : 0.0 - 50.0 Lux
Score de luminosité de la typologie : 0.0 %

Typologie étudiée : SejourCuisine / Intervalle : 1200.0 - 30000.0 Lux
Score de luminosité de la typologie : 6.0 %

Typologie étudiée : Loggia / Intervalle : 1200.0 - 30000.0 Lux
Score de luminosité de la typologie : 33.0 %

Score de luminosité total : 7.33 %

On peut voir que très peu de typologies respectent leurs critères respectifs, par exemple, seules 5% des cellules de la typologie chambre ne sont dans l'intervalle d'éclairement naturel modéré. On peut alors se demander, pourquoi ce critère n'est globalement pas respecté.

Voici ci-dessous, la répartition de la typologie chambre ainsi que la valeur de luminosité de chaque cellule :

Schéma de répartition :

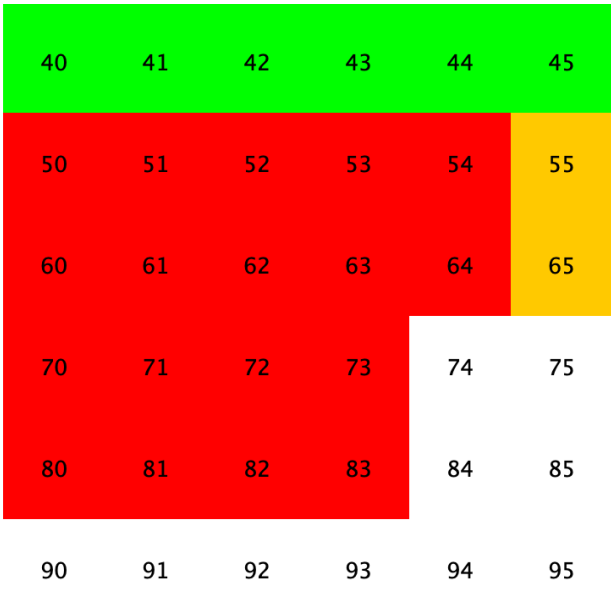


Schéma de luminosité :

0.0	0.0	0.0	0.0	48.0
0.0	0.0	0.0	82.0	1200.0
48.0	50.0	84.0	951.0	
1201.0	1248.0	1287.0	2184.0	

Cellules dans le bon intervalle

NOTE IMPORTANTE :

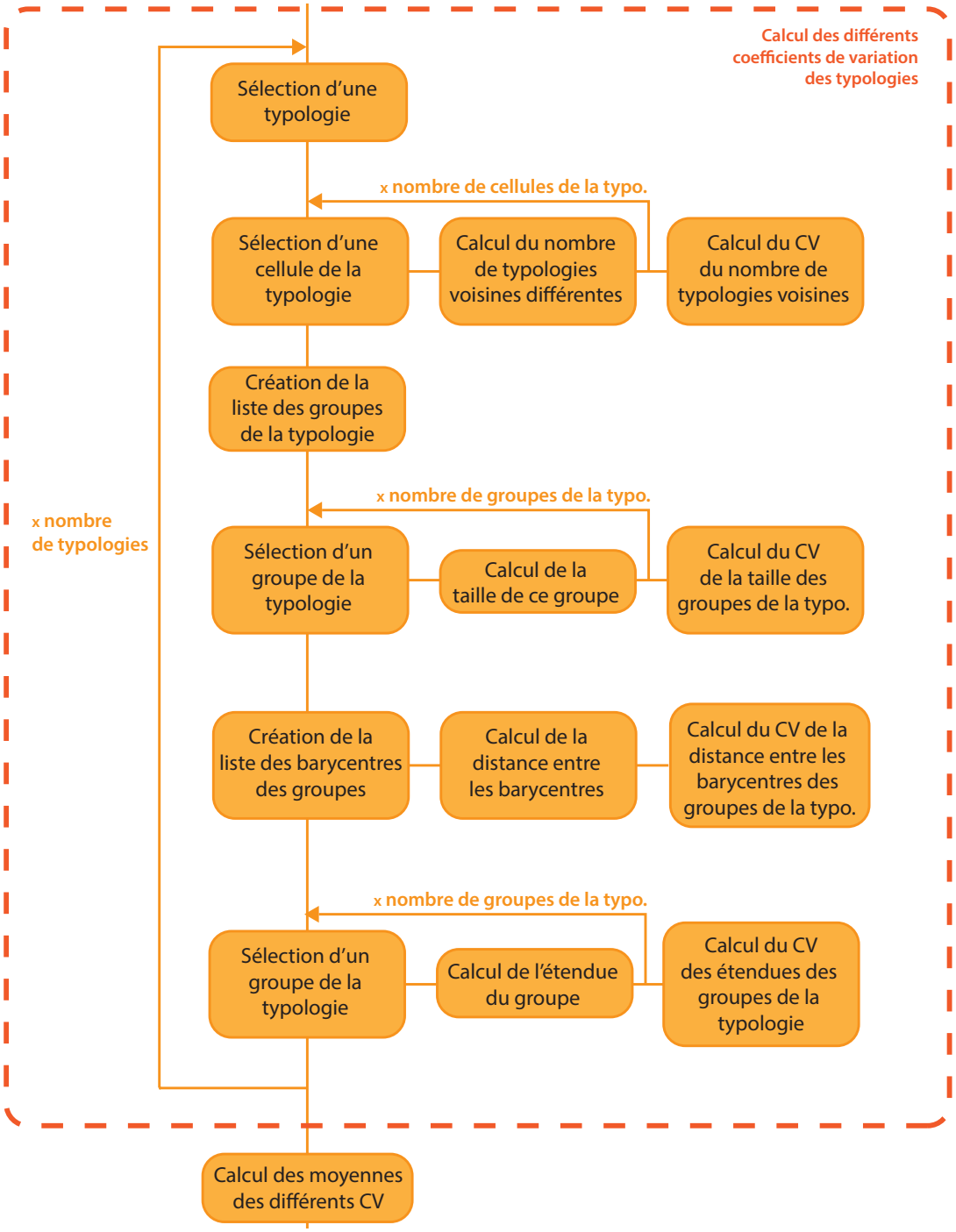
On peut voir que seulement deux cellules sont dans le bon intervalle. En analysant ce résultat de plus près, je me suis rendu compte d'une erreur commise lors de la mise en place de l'expérimentation : les cellules sont de taille 5m x 5m ce qui est beaucoup trop grand et influe directement sur le score de luminosité, le programme prenant en compte un modèle basé sur la distance. Malheureusement, le système de sauvegarde entre deux utilisations du programme ne concerne que les critères de modélisation et non les itérations (qui ne sont conservées que lors d'une utilisation), il m'est donc impossible de revenir sur cet élément en particulier. Voici le schéma de luminosité correcte calculé à la main basé sur une grille de 1m x 1m :

937	1041	2 379	3 937	3 750
2 605	3 334	5 417	11 041	15 000
5 156	6 250	9 375	26 250	30 000
16 406	17 500	20 625	37 500	30 000
30 000	30 000	30 000	30 000	30 000

On retrouve la valeur de 2 cellules dans le bon intervalle, mais avec des valeurs globalement trop élevées.

Ce pourcentage faible pour une typologie plutôt bien répartie démontre les limites d'un modèle simpliste dans le cas du calcul de la luminosité. Nous verrons par la suite comment améliorer ce modèle. Le calcul du score de luminosité des autres typologies relève du même principe.

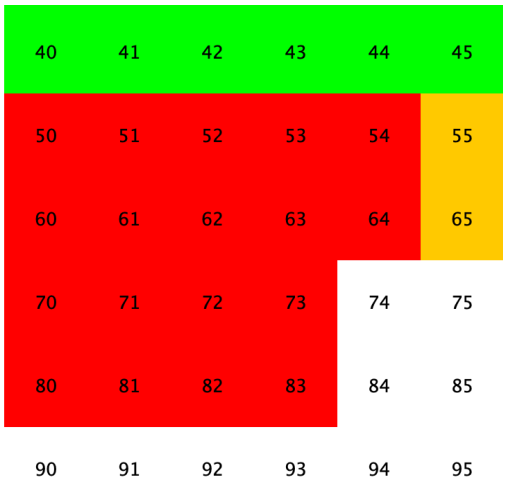
Nous allons à présent étudier le calcul du score typologique. Comme expliqué dans la partie précédente, celui-ci ne prend pas en compte le respect des critères imposés (nombre de cellules attribuées, etc.), ce qu'il faudrait rajouter. Le score typologique utilise principalement le calcul de la variance et de l'écart type afin de favoriser des tendances typologiques s'appuyant sur quatre critères : le nombre de typologies voisines différentes, la taille des groupes, la distance interne des groupes et la distance entre les groupes (on désigne par groupe un groupe de cellule).
Voici pour rappel un extrait de l'algorithme de calcul du score typologique :



Reprenons l'exemple de la typologie chambre, voici ce que donnent les logs :

Calcul du score de la typologie Chambre
Calcul du coefficient de variation des typologies voisines :
Coefficient de variation : 35.355339059327385
Calcul du coefficient de variation de taille de groupe :
Creation des différents groupes de typologie
Coefficient de variation : 0.0
Calcul du coefficient de variation de distance interne des groupes :
Coefficient de variation : 0.0

Les coefficients de variation sont des outils statistiques permettant de quantifier la variabilité d'un paramètre. Dans notre cas, on va chercher à créer des tendances et donc à uniformiser les paramètres, par exemple, dans le cas de logements universitaires, on voudra que les chambres soient de taille assez équivalente, qu'un maximum de chambre soit à proximité des lieux communs, etc. Cela permettra également de repérer les plans avec des cellules isolées ou des répartitions peu pratiques à utiliser.
On peut noter que les coefficients de variation concernant les groupes sont nuls, c'est tout simplement le cas lorsqu'il n'y a qu'un seul groupe. On peut penser qu'utiliser un coefficient de variation alors que ceci n'est pas nécessaire peut entrainer des problèmes, il ne faut cependant pas oublier qu'il s'agit ici de comparer différentes itérations entre elles, il est donc important de les considérer pour comparer les répartitions sur les mêmes critères.
Regardons comment est calculé le coefficient de variation du nombre de typologies voisines différentes :



Voici les différents cas possibles, le programme considère la cellule centrale et regarde les voisines :
À noter que le vide est considéré de typologie inconstructible plein

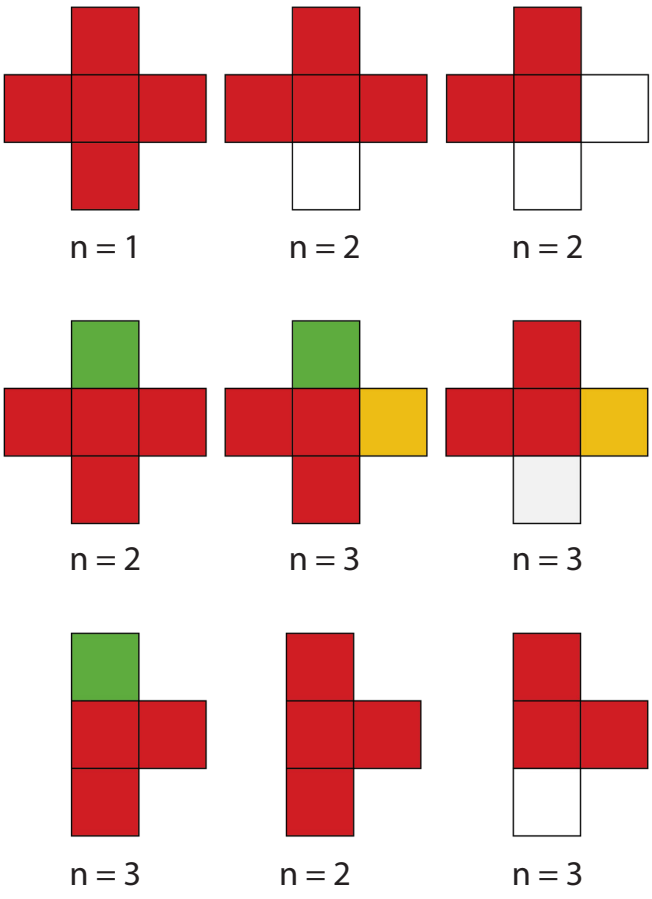


Diagramme des typologies voisines différentes

3	2	2	2	3
2	1	1	1	3
2	1	1	2	
3	2	2	2	

L'écart type est défini par la formule :

$$e = \sqrt{\frac{\sum |x - m|^2}{n}}$$

. $|x - m|^2$ vaut (en arrondissant) :
Pour 1 : 0,88 / pour 2 : 0,0036 / pour 3 : 1,12

. n : nombre de cellules = 18

Donc :
$\sum |x - m|^2$ = 1,12 x 4 + 0,0036 x 9 + 0,88 x 5

Et l'écart type peut être arrondi à : e = 0,7

On peut ainsi calculer le coefficient de variation :

$$cv = \frac{100 \times e}{m}$$ avec m la moyenne : m = 1,94

ainsi CV = 36,08
On retrouve approximativement le résultat du programme, l'approximation étant due aux arrondis.
On peut, de manière similaire, calculer les coefficients de variation pour chaque typologie. Le programme va ensuite faire la moyenne des quatre coefficients de variation puis la moyenne des quatre résultats obtenus. Plus le coefficient de variation est élevé plus on observe des disparités, le score va donc vouloir optimiser un petit coefficient de variation, il sera donc défini par :

Score typologique = 100 - Moyenne (CV) en pourcentage

Le score final sera alors la moyenne du score typologique et du score de luminosité.
De la même façon que pour la génération, on a obersvé quelques problèmes lors du calcul des résultats :

Problème 1 : Respect des consignes
Encore une fois, le calcul du score ne prend pas en compte du respect des critères, il serait intéressant d'implémenter cette fonction afin d'éviter des résultats ayant un bon socre mais ne répondant pas aux paramètres (typologies non générées, etc.)

Problème 2 : Calcul du score de luminosité
Le score de luminosité n'est visiblement pas adapté, et ce pour plusieurs raisons :
- Les valeurs varient énormément, il serait intéressant de revoir les intervalles. Ceux-ci ont pour l'instant été désignés de façon empirique au début du programme, il faudrait les ajuster avec

plusieurs essais. De même, demander à l'ensemble d'une typologie d'avoir la même luminosité n'a pas de sens, il peut être inutile d'avoir beaucoup de lumière dans les coins ou le fond de la pière, là où l'utilise moins.
- Le modèle est trop simpliste. En effet, la formule 1/x^2 ne fonctionne pas pour des distances inférieures à 1m. De plus, le programme calcul de cellule en cellule, pas directement des sources de luminosité, ce qui fausse les résultats. Enfin, un modèle de source de lumière ponctuelle n'est pas adapté dans ce cas précis, il faudrait utiliser un modèle à source de lumière surfacique ce qui demanderait de prendre en compte la hauteur des pièces et des fenêtres et donc de travailler en volume, ce qui n'est pas le cas actuellement.

Problème 3 : Redéfinition des coefficients de variations
L'outil coefficient de variation est très utile, mais demande de s'intéresser aux bonnes choses. Il pourrait être judicieux de revoir les paramètres pris en compte dans le calcul du score typologique, par exemple la notion de groupe de cellule n'a pas de sens dans le cas d'un seul bloc, on pourrait adapter le calcul en fonction de la typologie.

4) Retours sur l'expérimentation

Le premier constat suite à cette expérimentation est que le programme est loin d'être terminé et efficace. De nombreuses améliorations sont nécessaires que ce soit au niveau de la génération de l'itération, du calcul du score ou encore tout simplement au niveau de l'ergonomie du programme. Les origines de ces problèmes sont multiples, tout d'abord mon manque d'expérience en programmation entraînant de nombreuses erreurs, notamment au niveau de l'ergonomie ce qui ne facilite pas les expérimentations nécessaires au développement, mais également le fait d'avoir pensé ce programme sur presque deux ans, la vision de celui-ci ayant beaucoup changée durant ce laps de temps.
Certains résultats sont néanmoins prometteurs, ils sont parfois difficiles à déceler entre toutes les itérations, le système de score n'aidant pas à la tâche, mais ils mettent en avant des points positifs à exploiter. Le système de génération des typologies est intéressant, il mériterait un approfondissement des possibilités existantes, comme l'ajout de cases «aimants» et la correction de problèmes liés à l'algorithme, ainsi que le développement de nouveaux cas de figure. À mon sens, le système de score est quant à lui à revoir, le calcul du score typologique est relativement efficace et utile bien qu'incomplet, cependant le score de luminosité est à reprendre dans son intégralité. Pour que le programme soit plus performant, il faudrait également développer les autres scores possibles comme celui lié à l'acoustique ou aux déperditions énergétiques.
Le système de génération typologie par typologie présente des avantages comme le fait de hiérarchiser celles-ci, néanmoins il est source d'erreur et cela mériterait de se pencher vers d'autres solutions possibles. L'une d'elles serait de passer à un système générant simultanément les typologies ou tout du moins une partie de la typologie à la fois ce qui permettrait de mieux répartir les typologies dans l'espace. Une seconde solution serait de passer l'itération obtenue à un crible d'affinage permettant de corriger celui-ci en intervertissant des cellules par exemple.
Les différents problèmes non corrigés ainsi que l'état non terminé du programme ne m'ont pas laissé l'occasion de me poser la question d'optimisation, point qui serait inévitable étant donné que le nombre d'itérations influe directement sur la qualité du résultat. Le programme est relativement rapide pour une centaine de plans, mais il faut plus d'une minute pour la génération de 10 000 plans relativement simples pour une grille qui n'est finalement pas très grande. Étant donné

qu'il s'agit d'une boucle pour chaque itération, il serait plutôt facile de réduire le temps alloué à chacune d'entre elles et ainsi de diminuer drastiquement la durée de génération.

Conclusion

Le sujet abordé par ce mémoire était l'utilisation et la transformation de critères en esquisses de plan, et ce via la création et l'utilisation d'un logiciel génératif. Cet exercice a été pour moi une excellente occasion de m'initier à un processus de recherche, de découvrir et d'approfondir un sujet par mes propres moyens, de me baser sur les travaux de chercheurs ou d'autres étudiants et d'interpréter leurs résultats afin de proposer une alternative à des problèmes posés depuis bien longtemps.

La question que l'on peut se poser après la lecture de ce mémoire et l'étude des résultats obtenus est : est-il finalement possible de faire un programme qui fonctionne et à fortiori d'utiliser des systèmes de génération en architecture ? Comme expliqué dans la partie concernant l'État de l'art historique et actuel, c'est une problématique qui remonte aux années 70 puis qui a peu à peu laissé place à l'architecturologie de par sa nature contradictoire basée sur des compromis qu'un programme ne saurait résoudre. Je pense cependant qu'une approche différente est possible. Il me semble compliqué, voir irréaliste, de demander à un programme de jouer le rôle d'un architecte et quand bien même ce serait possible, l'architecture est trop diverse pour être cantonnée à des lignes de codes décrivant essentiellement des notions d'efficacité et d'optimisation, néanmoins il serait possible de demander à un programme de dessiner un panel d'esquisses qui, affiné par l'architecte, pourrait servir de fondations à un projet architectural. C'est justement ce qui manque au programme présenté dans ce mémoire, ce principe de parcourir l'ensemble des possibilités et de les présenter à l'utilisateur (notion non incorporée dans le programme, mais tout de même présentée dans la partie «Les bases du programme») et pour une raison bien simple, ce n'est pas le but que je visais au départ. L'idée de départ était de voir si un programme pouvait créer le plan parfait, et en plus d'un an et demi de développement, l'idée a eu le temps de germer et de grandir dans d'autres directions. Si le mémoire était à refaire, je le referais différemment. C'est un peu facile d'affirmer cela maintenant après avoir acquis cette vision des choses suite, justement, au travail et aux recherches effectués pour ce mémoire.
Le concept principal menant à ce constat est celui de l'apprentissage. Actuellement, une itération du programme est soit parfaite soit inutile, rien n'est appris d'une esquisse imparfaite et c'est ce principe que viendrait corriger l'utilisation d'arborescences menant à un panel contrôlé par l'utilisateur. L'architecte, en remontant dans l'arbre de création de l'itération pourrait alors explorer de nombreuses autres branches de possibilité et cueillir le meilleur fruit pour rester dans la métaphore dendrologique.

Pour conclure, puisqu'il s'agit d'une conclusion, l'utilisation d'un programme permettant d'aiguiller les principes architecturaux (esquisses de plan) d'un projet à l'aide de critères définis et paramétrés par l'architecte est possible et pourrait être une bonne idée, mais dans tous les cas il serait dommageable d'ignorer cette possibilité ainsi que ce qu'elle a à offrir.

Bibliographie

Agraa, O.M., Whitehead, B., 1968. **Nuisance restrictions in the planning of single-storey layouts**. Building Science 2, 291–302

Alexander, C., 1964. **Notes on the synthesis of form**, 17. printing. ed. Harvard Univ. Press, Cambridge, Mass.

Alexander, C., 1963. **HIDECS 3: Four computer programs for the hierarchical decomposition of systems which have an associated linear graph**. MIT.

Bielefeld, B. (Ed.), 2015. **Planning architecture: dimensions and typologies**. Birkhäuser Verlag GmbH, Basel.

Boudier, J.-P., Charalambides, S., Fourcade, A.-M., Lafue, G., 1973. **Analyse de programme d'allocation spatiale**. Presented at the Séminaire sur l'allocation spatiale, Institut de l'environnement, Paris.

Buxton, P., 2018. **Metric handbook: planning and design data**.

CERVELLE-J, 2010. **Complexité dynamique et algorithmique des automates cellulaires**. OMNISCRIPTUM, S.l.

Delahaye, J.-P., 2009. **Le royaume du Jeu de la vie** [WWW Document]. Pourlascience.fr. URL https://www.pourlascience.fr/sd/mathematiques/le-royaume-du-jeu-de-la-vie-2944.php

Fredet, J., 2016. **Mettre en forme et composer le projet d'architecture: diverses considérations sur les manières de procéder, héritées puis réformées par les avant-gardes du XXe siècle avec perspectives d'évolution au début du suivant**, vol. 1-2. Fario, Paris.

Hafez, E., Agraa, O.M., Whitehead, B., 1967. **Automation of data preparation in computer programme for the planning of single-storey layouts**. Building Science 2, 83–88.

Hossbach, B., Eichelmann, C., Lehmhaus, C., 2016. **Competition panels and diagrams: construction and design manual**. DOM publishers, Berlin.

Langlois, A., Phipps, M., 1997. **Automates cellulaires: application à la simulation urbaine**. Hermès, Paris.

Levin, P.H., Building Research Station (Great Britain), 1964. **Use of graphs to decide the optimum layout of buildings**. Building Research Station, Garston.

Negroponte, N.P., 1966. **The computer simulation of perception during motion in the urban environment**. (Thesis). Massachusetts Institute of Technology.

Parslow, R.D., Green, R.E., 1971. **Advanced Computer Graphics: Economics Techniques and Applications**. Springer US, Boston, MA.

Picon, A., 2010. **Digital culture in architecture: an introduction for the design professions**. Birkhäuser, Basel.

ScienceEtonnante, 2017. **Le Jeu de la Vie — Science étonnante #49**.