
Tiled Documentation

Version 1.7.2

Thorbjørn Lindeijer

oct. 27, 2021

Manuel d'Utilisation

1	Introduction	3
1.1	A propos de Tiled	3
1.2	Pour Débuter	4
2	Projets	9
2.1	Ce qu'il y a dans un Projet	9
2.2	Sessions	9
2.3	Ouvrir un Fichier dans le Projet	10
3	Travailler avec des Calques	11
3.1	Types de Calques	12
3.2	Facteur de Défilement de Parallaxe	13
3.3	Teindre des Calques	13
4	Modifier des Calques de Tuiles	15
4.1	Brosse Tampon	15
4.2	Brosse de Terrain	16
4.3	Outil de Seau de Remplissage	16
4.4	Outil de Remplissage de Forme	16
4.5	Gomme	17
4.6	Outils de Sélection	17
4.7	Gérer les Tampons de Tuile	17
5	Travailler avec des Objets	19
5.1	Outils de Placement	19
5.2	Sélectionner des Objets	21
5.3	Éditer des Polygones	23
5.4	Connecter des Objets	23
6	Modifier des Jeux de Tuiles	25
6.1	Deux Types de Jeux de Tuiles	25
6.2	Propriétés des Jeux de Tuiles	26
6.3	Propriétés des Tuiles	26
6.4	Information de Terrain	27
6.5	Éditeur de Collision de Tuiles	27
6.6	Éditeur d'Animation de Tuile	28

7 Propriétés Personnalisées	31
7.1 Ajout de Propriétés	32
7.2 Héritage des Propriétés de Tuile	32
7.3 Propriétés Prédéfinies	32
8 Utiliser des Modèles	35
8.1 Créer des Modèles	36
8.2 Créer des Instances de Modèle	36
8.3 Éditer des Modèles	36
8.4 Détacher des Instances de Modèle	37
9 Utiliser des Terrains	39
9.1 Définir les Informations de Terrain	40
9.2 Édition avec la Brosse de Terrain	42
9.3 Mode de Remplissage de Terrain	43
9.4 Probabilité de Tuile et de Terrain	44
9.5 Transformations de Tuile	45
9.6 Derniers Mots	45
10 Utiliser des Cartes Infinies	47
10.1 Créer une Carte Infinie	48
10.2 Éditer la Carte Infinie	48
10.3 Conversion d'une Carte Infinie vers une Carte Finie et Vice Versa	48
11 Travailler avec des Mondes	53
11.1 Définir un Monde	54
11.2 Éditer des Mondes	54
11.3 Utiliser la Correspondance de Motif	55
11.4 Seulement Montrer les Voisins Directs	56
12 Utiliser les Commandes	57
12.1 Le Bouton de Commande	57
12.2 Éditer les Commandes	57
12.3 Commandes d'Exemple	58
13 Automapping	59
13.1 Qu'est-ce que l'Automapping ?	59
13.2 Mise en Place	59
13.3 Exemples	62
14 Exporter des Formats	81
14.1 Formats de Fichiers Génériques	81
14.2 Defold	82
14.3 GameMaker : Studio 1.4	83
14.4 GameMaker Studio 2.3	85
14.5 tBIN	90
14.6 Autres Formats	90
14.7 Formats d'Exportation Personnalisés	91
14.8 Exporter en Tant qu'Image	92
15 Raccourcis Clavier	93
15.1 Général	93
15.2 Quand un calque de tuiles est sélectionné	94
15.3 Quand un calque d'objets est sélectionné	95
15.4 Dans la Boite de Dialogue de Propriétés	95

16	Préférences Utilisateur	97
16.1	Général	97
16.2	Interface	99
16.3	Clavier	101
16.4	Thème	102
16.5	Greffons	103
17	Scripts Python	105
17.1	Exemple de Greffon d'Exportation	106
17.2	Déboguer Votre Script	107
17.3	Référence de l'API	107
18	Bibliothèques et Environnements	109
18.1	Support par Langage	109
18.2	Support par Environnement	112
19	Format de Carte TMX	119
19.1	<map>	120
19.2	<editorsettings>	121
19.3	<tileset>	121
19.4	<layer>	124
19.5	<objectgroup>	126
19.6	<imagelayer>	128
19.7	<group>	128
19.8	<properties>	128
19.9	Fichiers de Modèle	129
20	Notes de Version de TMX	131
20.1	Tiled 1.7	131
20.2	Tiled 1.5	131
20.3	Tiled 1.4	132
20.4	Tiled 1.3	132
20.5	Tiled 1.2.1	132
20.6	Tiled 1.2	132
20.7	Tiled 1.1	132
20.8	Tiled 1.0	133
20.9	Tiled 0.18	133
20.10	Tiled 0.17	133
20.11	Tiled 0.16	133
20.12	Tiled 0.15	133
20.13	Tiled 0.14	134
20.14	Tiled 0.13	134
20.15	Tiled 0.12	134
20.16	Tiled 0.11	134
20.17	Tiled 0.10	134
20.18	Tiled 0.9	135
20.19	Tiled 0.8	136
21	Format de Carte JSON	137
21.1	Carte	138
21.2	Calque	140
21.3	Fragments	142
21.4	Objet	142
21.5	Texte	146
21.6	Jeu de Tuiles	147

21.7	Modèle d'Objet	152
21.8	Propriété	152
21.9	Point	152
21.10	Notes de Version	152
22	Global Tile IDs	155
22.1	Tile Flipping	155
22.2	Mapping a GID to a Local Tile ID	156
22.3	Code example	156
23	Création de Script	159
23.1	Introduction	159
23.2	Référence de l'API	160

Note : Si vous ne trouvez pas ce que vous cherchez dans ces pages, n'hésitez pas à poser des questions sur le [Forum de Tiled](#) ou sur le [serveur Discord de Tiled](#).

CHAPITRE 1

Introduction

1.1 A propos de Tiled

Tiled est un éditeur de niveau en 2D qui vous aide à développer le contenu de votre jeu. Sa principale caractéristique est d'éditer des cartes de tuile aux formes variées, mais prend également en charge le placement d'image libre ainsi que des moyens efficaces d'annoter votre niveau avec des informations supplémentaires utilisées par le jeu. Tiled se concentre sur la flexibilité générale en essayant de rester intuitif.

En termes de cartes de tuile, il supporte les calques de tuiles rectangulaire droite, mais également les calques de tuiles isométriques, isométrique en quinconce et hexagonal en quinconce. Un jeu de tuiles peut être soit une seule image contenant plusieurs tuiles, ou peut aussi être une collection d'images individuelles. Afin de supporter certaines techniques de simulation de profondeur, les tuiles et les calques peuvent être décalés par une distance personnalisée et leur ordre de rendu peut être configuré.

L'outil principal pour éditer les *calques de tuiles* est un pinceau tampon qui permet un remplissage et une copie efficace de zones de tuiles. Il supporte aussi le dessin de lignes et de cercles. De plus, il y a plusieurs outils de sélection et un outil qui fait des *transitions automatiques de terrain*. Finalement, il peut appliquer des changements basés sur une *correspondance de motif* pour automatiser des parties de votre travail.

Tiled supporte aussi des *calques d'objets*, qui étaient auparavant seulement utilisés pour annoter votre carte avec des informations, mais plus récemment ils peuvent également être utilisés pour placer des images. Vous pouvez ajouter des objets rectangulaires, elliptiques, polygonaux, des polylinéaires et des tuiles objets. Le placement des objets n'est pas limité à la grille de tuile et ceux-ci peuvent aussi être redimensionnés ou pivotés. Les calques d'objets offrent beaucoup de flexibilité pour ajouter presque n'importe quelle information à votre niveau dont votre jeu a besoin.

D'autres choses qui valent d'être mentionnées sont le support d'ajout de format de jeu de tuiles ou de carte personnalisés au travers de greffons, l'*extension de Tiled* via JavaScript, la mémoire du tampon de tuile, le *support d'animation de tuile* et l'*éditeur de collision de tuile*.

1.2 Pour Débuter

1.2.1 Mettre en Place un Nouveau Projet

Lors du premier lancement de Tiled, nous sommes accueillis par cette fenêtre suivante :

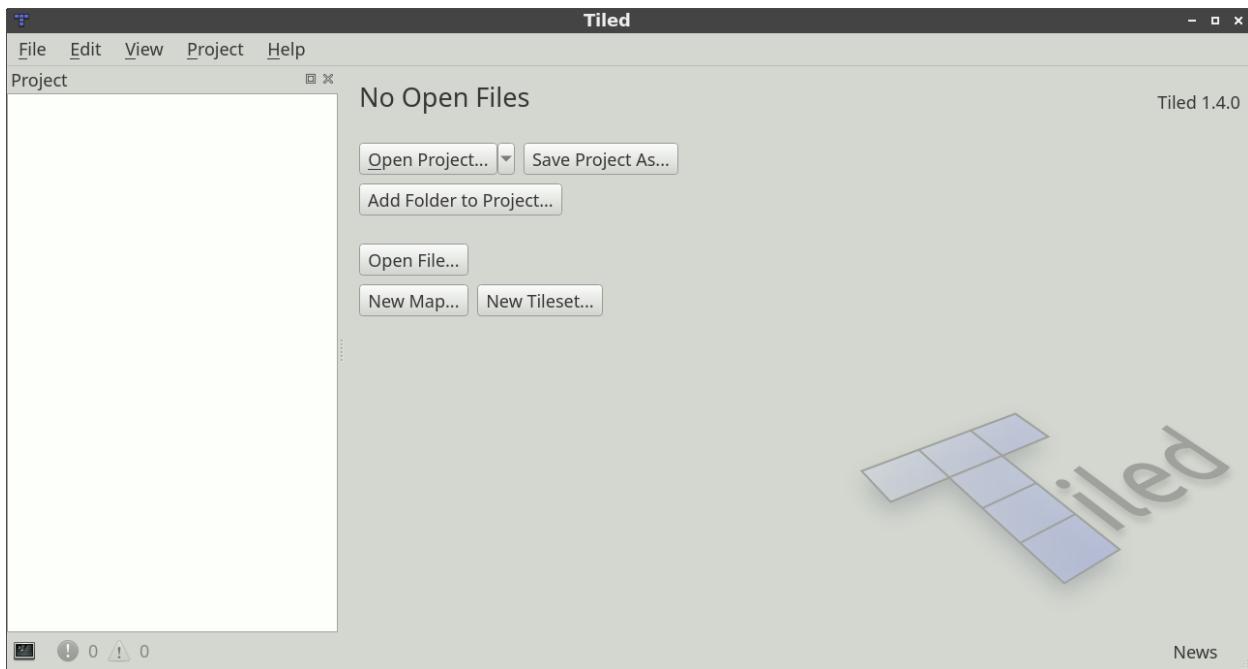


Fig. 1 – Fenêtre de Tiled

Pour que toutes nos ressources soient facilement accessibles depuis la vue *Projet* ainsi que pour être capable de passer d'un projet à un autre rapidement, il est recommandé de d'abord mettre en place un projet Tiled. C'est cependant une étape complètement optionnelle qui peut être passée si vous le voulez.

Choisissez *Projet -> Sauver le Projet en tant que...* pour sauvegarder un nouveau fichier de projet. La location recommandée est la racine de votre projet, mais vous pouvez le placer où vous voulez.

Ensuite, nous allons ajouter au moins un dossier, que ce soit un dossier de « ressources » ou simplement la racine de votre projet, mais vous pouvez aussi choisir d'ajouter plusieurs dossiers de haut niveau tels que « tilesets » (jeux de tuiles), « maps » (cartes), « templates » (modèles), etc. Faites un clic droit sur la vue *Projet* et choisissez *Ajouter un Dossier au Projet...* pour ajouter les dossiers utiles.

1.2.2 Créer une Nouvelle Carte

Pour créer une nouvelle carte, choisissez *Fichier -> Nouveau -> Nouvelle Carte...* (*Ctrl+N*). La boîte de dialogue suivante va apparaître :

Ici, nous choisissons la taille initiale de la carte, la taille de tuile, l'orientation, le format des calques de tuiles et l'ordre de rendu des tuiles (seulement supporté pour les cartes *Orthogonales*) et si la carte est *infinie* ou non. Toutes ces choses peuvent être changées plus tard si nécessaire, donc ce n'est pas important de tout bien faire du premier coup.

Note : Si vous créez un projet, faites en sorte de sauvegarder la carte dans un dossier que vous avez ajouté à votre projet. Cela va la rendre facilement accessible en utilisant *Fichier -> Ouvrir un Fichier dans le Projet* (*Ctrl+P*).

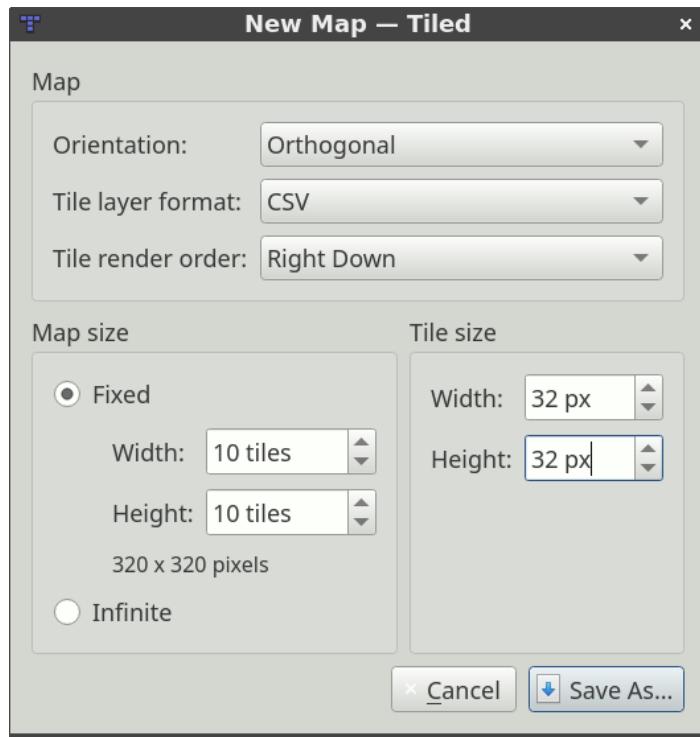


Fig. 2 – Nouvelle Carte

Après avoir sauvegardé notre carte, nous verrons la grille de tuile et un calque de tuiles initial sera ajouté à la carte. Toutefois, avant de pouvoir commencer à utiliser des tuiles nous devons ajouter un jeu de tuile. Choisissez *Carte -> Nouveau -> Nouveau Jeu de Tuiles...* pour ouvrir la boîte de dialogue de Nouveau Jeu de Tuile :

Cliquez sur le bouton *Parcourir...* et sélectionnez le `tmw_desert_spacing.png` jeu de tuiles à partir des exemples fournis avec Tiled (ou utilisez un des vôtres si vous le souhaitez). Cet exemple de jeu de tuiles utilise des tuiles de taille 32x32. Il y a aussi une *marge* d'un pixel autour de la tuile et un *espacement* d'un pixel entre les tuiles (c'est assez rare normalement, vous devrez généralement laisser ces valeurs à 0).

Note : Nous laissons l'option *Embarquer dans la carte* désactivée. Ceci est recommandé, car il va permettre l'utilisation du jeu de tuiles par plusieurs cartes sans avoir à donner ses propriétés une nouvelle fois. C'est aussi une bonne chose de stocker le jeu de tuiles dans son propre fichier si vous voulez par la suite ajouter des propriétés de tuiles, des définitions de terrain, des formes de collision, etc., car cette information est partagée entre toutes vos cartes.

Après avoir sauvegardé le jeu de tuiles, Tiled devrait ressembler à ceci :

Comme nous ne voulons pas faire grand chose de plus avec le jeu de tuiles pour le moment, revenez sur le fichier de carte :

Nous sommes prêts à sélectionner des tuiles et commencer l'édition ! Mais d'abord, jetons un coup d'œil aux *differents types de calques* pris en charge par Tiled.

Note : La plupart du manuel doit encore être écrit. Heureusement, il y a une très bonne *Série Tutoriel sur l'Éditeur de Carte Tiled* sur GamesFromScratch.com. En outre, le support pour Tiled dans divers *moteurs et environnements* vient souvent avec quelques informations d'utilisation.

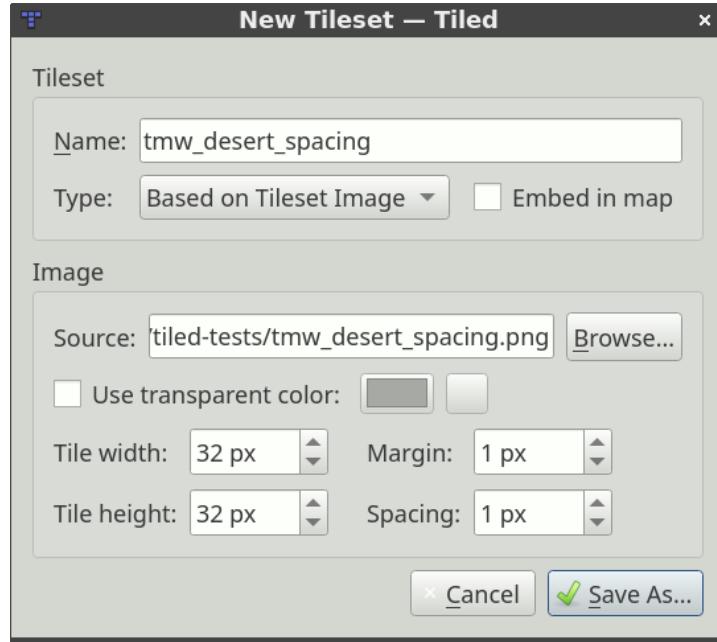


Fig. 3 – Nouveau Jeu de Tuile

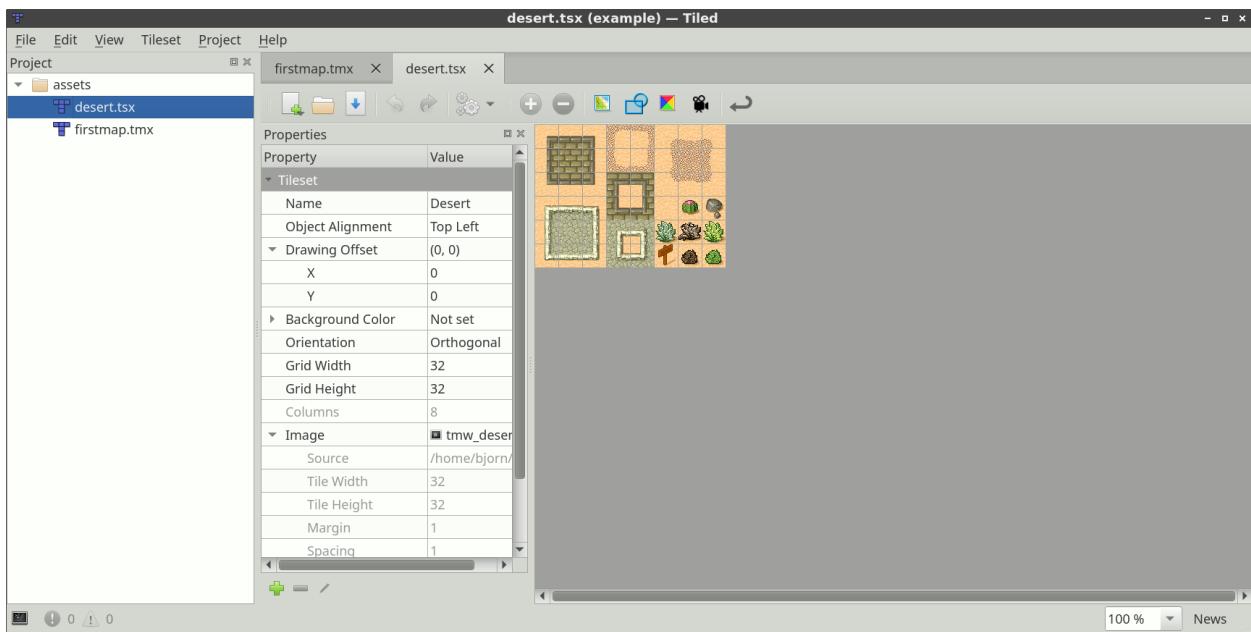


Fig. 4 – Jeu de Tuiles Crée

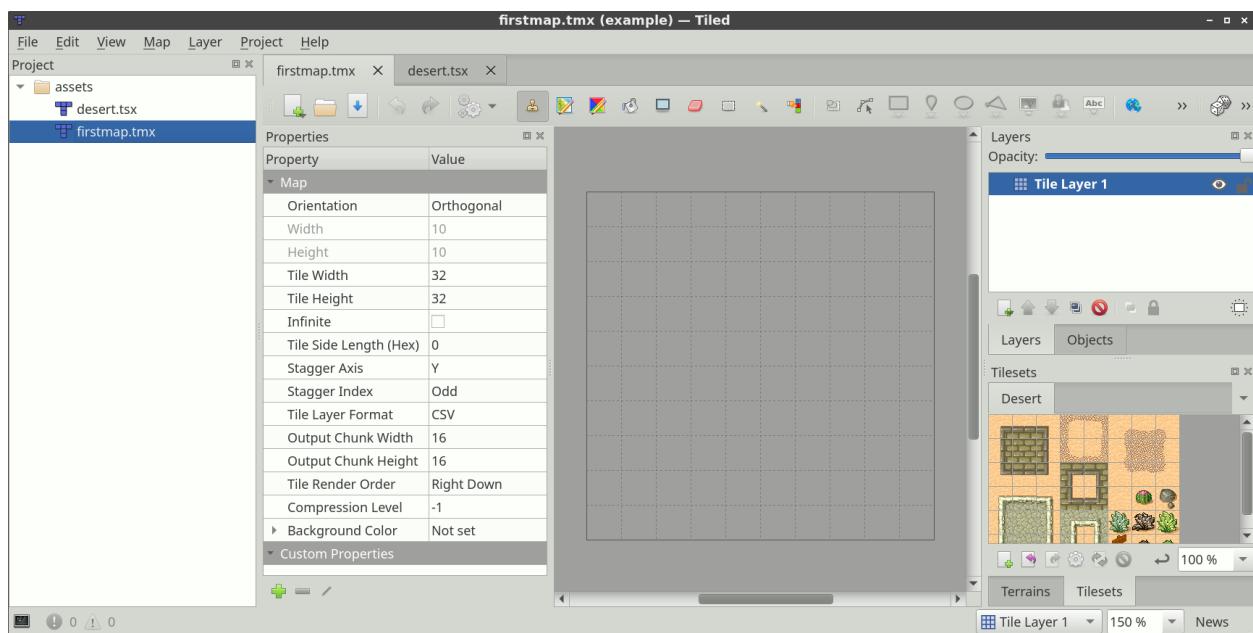


Fig. 5 – Jeu de Tuiles Utilisable sur la Carte

CHAPITRE 2

Projets

2.1 Ce qu'il y a dans un Projet

Un projet Tiled définit la liste des dossiers contenant les ressources appartenant à ce projet en premier lieu. De plus, il fournit une base pour le *fichier de session*.

Mis à part la liste de dossiers, un projet comprend les propriétés suivantes, qui peuvent être changées dans la boîte de dialogue *Projet -> Propriétés de Projet...*.

Dossier d'Extensions Un dossier spécifique au projet dans lequel vous pouvez mettre des *extensions Tiled*. Sa valeur est simplement **extensions** par défaut, donc si vous avez un dossier nommé « extensions » à côté de votre fichier de projet, il sera choisi automatiquement.

Le dossier est chargé en plus des extensions globales.

Fichier de Types d'Objet Réfère au fichier définissant les types d'objet. Faites attention à spécifier ce fichier avant d'ouvrir *l'Éditeur de Types d'Objet*, pour être sûr que tous les types que vous avez définis sont sauvegardés au bon endroit.

Lorsque cette option n'est pas spécifiée, les définitions de types d'objet sont sauvegardées dans une location globale.

Fichier de Règles d'Automapping Réfère à un fichier de règles d'automapping qui doit être utilisé pour toutes les cartes tant que ce projet est chargé. Ignoré pour les cartes qui ont un fichier ``rules.txt`` sauvegardé à côté d'elles.

2.2 Sessions

Chaque fichier de projet a un fichier *tiled-session* associé stocké avec lui. Le fichier de session ne doit généralement pas être partagé avec d'autres personnes et stocke vos derniers fichiers ouverts, une partie de leur dernier état dans l'éditeur, les derniers paramètres utilisés dans les boîtes de dialogue, etc.

Lors d'un changement de projet Tiled passe à la session associée automatiquement, pour que vous puissiez facilement reprendre là où vous vous êtes arrêtés. Quand aucun projet n'est chargé, un fichier de session global est utilisé.

2.3 Ouvrir un Fichier dans le Projet

Un autre avantage de la mise en place d'un projet est que vous pouvez rapidement ouvrir n'importe quel fichier avec une extension reconnue dans un des dossiers du projet. Utilisez *Fichier -> Ouvrir un Fichier dans le Projet* (Ctrl + P) pour ouvrir le filtre de fichiers et juste entrer le nom du fichier que vous voulez ouvrir.

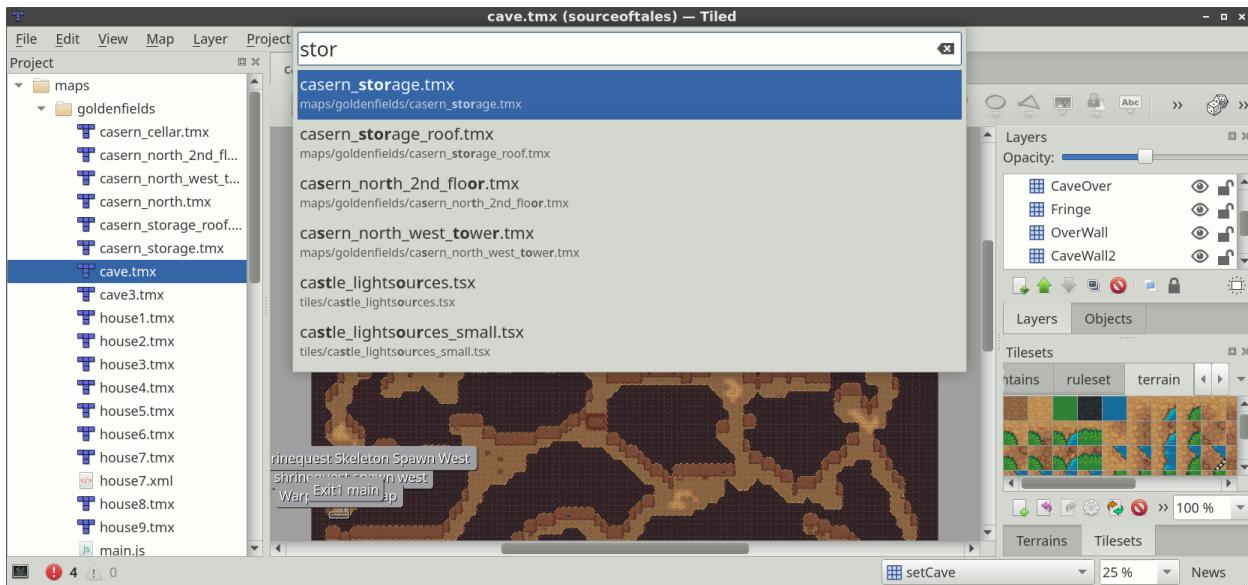


Fig. 1 – Ouvrir un Fichier dans le Projet

Futures Extensions

Il y a plusieurs moyens par lesquels les projets peuvent être rendus plus puissants :

- Rendre le projet accessible depuis [l'API de création de script](#).
- Autoriser la désactivation de fonctionnalités sur une base par projet, pour simplifier l'interface utilisateur et réduire les chances d'accidentellement faire quelque chose que votre projet ne supporte pas.
- Reconnaître les différentes ressources de votre projet pour que la sélection d'images, de jeux de tuiles et de modèles puisse être plus efficace (potentiellement en remplaçant la boîte de dialogue de fichiers système).

Si vous aimez au moins un seul de ces points, s'il-vous-plaît aidez-moi à y arriver plus vite en [supportant le développement de Tiled](#). Plus je recevrai de soutien, plus j'aurais les moyens de passer du temps à améliorer Tiled !

CHAPITRE 3

Travailler avec des Calques

Une carte Tiled supporte des types de contenus variés, et ce contenu est organisé dans de différents calques variés. Les calques les plus communs sont les *Calques de Tuiles* et les *Calques d'Objets*. Il existe aussi des *Calques d'Images* pour inclure de simples graphismes d'avant-plan ou d'arrière-plan. L'ordre des calques détermine l'ordre de rendu de votre contenu.

Les calques peuvent être cachés, rendus partiellement visible et peuvent être verrouillés. Les calques ont aussi un décalage et un *facteur de défilement de parallaxe*, qui peut être utilisé pour les disposer de façon indépendante des autres, pour par exemple simuler de la profondeur. Finalement, leur contenu peut être teinté en le multipliant par une *couleur de teinte* personnalisée.

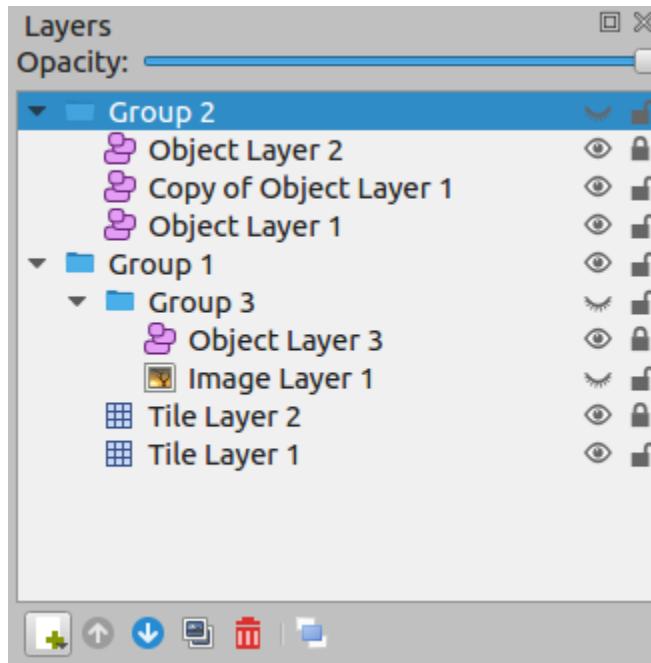


Fig. 1 – L'icône d'œil et de cadenas activent/désactivent respectivement la visibilité et l'état de verrouillage du calque.

Vous pouvez utiliser des *Groupes de Calques* pour organiser les calques dans une hiérarchie. Cela permet de rendre le travail avec beaucoup de calques plus confortable.

3.1 Types de Calques

3.1.1 Calques de Tuiles

Les calques de tuiles fournissent un moyen efficace de stocker une grande aire remplie de données de tuiles. Ces données sont un simple tableau de références de tuiles et ne peuvent donc pas stocker d'informations additionnelles pour chaque location. Les seules données supplémentaires qui sont stockées sont quelques drapeaux, qui permettent au graphismes des tuiles d'être inversés verticalement, horizontalement ou anti-diagonalement (pour supporter la rotation par incrément de 90 degrés).

L'information requise pour générer un rendu pour chaque calque de tuiles est stocké dans la carte, qui spécifie la position et l'ordre de rendu des tuiles basé sur l'orientation et des propriétés de tuiles diverses.

Même s'ils ne peuvent que référer des tuiles, les calques de tuiles peuvent être utiles pour définir des morceaux d'informations non-graphiques dans votre niveau. Les informations de collision peuvent souvent être portées en utilisant un jeu de tuiles spécial, et tout objet qui n'a pas besoin de propriété personnalisée et qui est toujours aligné avec la grille peut aussi être placé sur un calque de tuiles.

3.1.2 Calques d'Objets

Les calques d'objets sont utiles car ils peuvent stocker beaucoup de types d'informations qui ne rentrent pas dans un calque de tuiles. Les objets peuvent être positionnés librement, redimensionnés et pivotés. Ils peuvent aussi avoir des propriétés personnalisées individuelles. Il y a plein de types d'objets :

- **Rectangle** - pour baliser des aires rectangulaires
- **Ellipse** - pour baliser des aires d'ellipse ou circulaires personnalisées
- **Point** - pour baliser des locations exactes (depuis Tiled 1.1)
- **Polygone** - pour quand un rectangle ou un cercle n'est pas suffisant (souvent une aire de collision)
- **Polyligne** - peut être un chemin à suivre ou un mur de collision
- **Tuile** - pour placer, redimensionner et tourner librement votre graphisme de tuile
- **Texte** - pour des notes ou un texte personnalisé (depuis Tiled 1.0)

Tous les objets peuvent être nommés, ce qui va montrer le nom de l'objet dans une étiquette au-dessus de lui-même (seulement activé pour les objets sélectionnés par défaut). Les objets peuvent aussi recevoir un *type*, ce qui est utile car cela permet de personnaliser la couleur de leur étiquette et les *propriétés personnalisées* de ce type d'objet. Pour les objets tuile, le type peut être *hérité de leur tuile*.

Pour la majorité des types de cartes, les objets sont positionnées dans des pixels entiers. La seule exception à ceci sont les cartes isométriques (pas isométriques échelonnées). Pour les cartes isométriques, il a été jugé utile de stocker ses positions dans des coordonnées spatiales projetées. Pour ceci, les tuiles isométriques sont supposées représenter des carrés projetés avec les deux côtés égaux à la *hauteur de la tuile*. si vous utilisez un espace coordonné différent pour les objets dans votre jeu isométrique, vous aurez besoin de convertir ces coordonnées respectivement.

La longueur et la hauteur de l'objet est aussi stockés majoritairement en tant que pixels. Pour les cartes isométriques, tous les objets de formes (rectangle, point, ellipse, polygone et polyligne) sont projetés dans le même espace coordonné décrit ci-dessus. C'est basé sur l'hypothèse que ces objets sont généralement utilisés pour baliser des aires sur la carte.

3.1.3 Calques d'Images

Les calques d'images fournissent un moyen d'inclure une seule image en tant qu'avant-plan ou arrière-plan de votre carte rapidement. Ils ne sont pour l'instant pas très utiles, parce que si vous voulez à la place ajouter l'image en tant que Jeu de Tuiles et la placer en tant qu'"*Objet Tuile*", vous avec la possibilité de redimensionner et de pivoter l'image librement.

Le seul avantage à utiliser un calque d'images est qu'il évite la sélection / le glissement de l'image lors de l'utilisation de l'outil de Sélection d'Objets. Cependant, depuis Tiled 1.1, cela peut aussi être fait en verrouillant le calque d'objets contenant l'objet tuile avec lequel vous voulez éviter d'interagir.

3.1.4 Groupes de Calques

Les groupes de calques marchent comme des dossiers et peuvent être utilisés pour organiser les calques dans une hiérarchie. C'est surtout utile lorsque votre carte contient un grand nombre de calques.

La visibilité, l'opacité, le décalage, le verrouillage et la *couleur de teinte* d'un groupe de calques affectent tous ses calques enfants.

Les calques peuvent être facilement glissés dans ou en dehors de groupes avec la souris. Les actions Monter le Calque / Descendre le Calque vous permettent aussi de déplacer les calques dans ou en dehors des groupes.

3.2 Facteur de Défilement de Parallaxe

Le facteur de défilement de parallaxe détermine la distance par laquelle le calque bouge par rapport à la caméra.

Par défaut sa valeur est 1, ce qui veut dire que sa position sur l'écran change à la même vitesse que la position de la caméra (dans la direction opposée). Une valeur plus petite le fera bouger plus lentement, simulant un calque qui est éloigné, là où une valeur plus grande le fera bouger plus rapidement, simulant un calque entre l'écran et la caméra.

Une valeur de 0 ne fait pas bouger le calque du tout, ce qui peut être utile pour inclure des éléments de votre interface graphique en jeu ou pour baliser les bords de la fenêtre d'affichage générale.

Des valeurs négatives bougent le calque dans la direction opposée, cependant c'est rarement utile.

Quand le facteur de défilement de parallaxe est donné sur un groupe de calques, il s'applique à tous ses calques enfants. Le facteur de défilement de parallaxe effectif est déterminé en multipliant le facteur de défilement de parallaxe par le facteur de défilement de tous les calques parents.

3.3 Teindre des Calques

Quand vous définissez la propriété *Couleur de Teinte* d'un calque, cela affecte la façon dont les calques sont rendus. Cela inclut les tuiles, les objets tuiles et les images dans un *Calque d'Images*.

Chaque valeur de couleur de pixel est multipliée par la couleur de teinte. De cette façon vous pouvez assombrir ou colorier vos graphismes de plusieurs façons sans avoir besoin de mettre en place des images séparées pour cela.

La couleur de teinte peut aussi être définie sur un *Groupe de Calques*, et dans ce cas elle est héritée par tous les calques de ce groupe.

Futures Extensions

Il y a plusieurs moyens par lesquels les calques peuvent être rendus plus puissants :

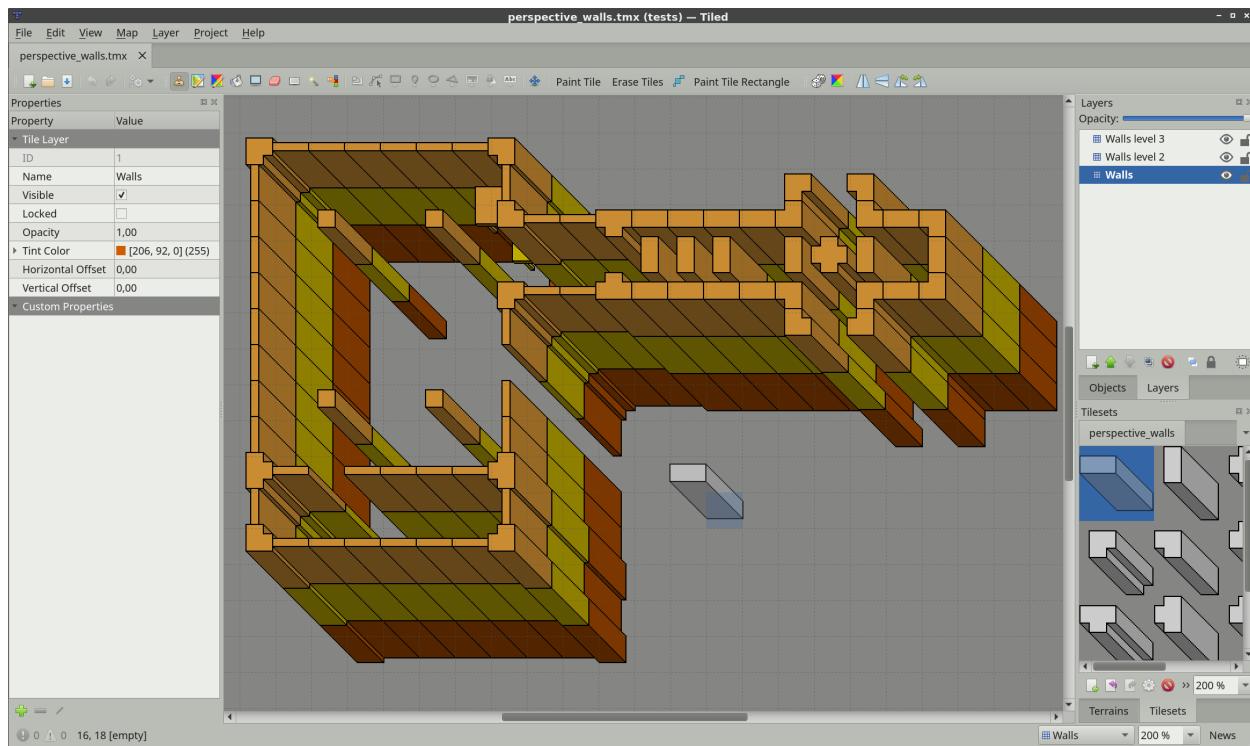


Fig. 2 – Un jeu de tuiles gris rendu dans une couleur différente pour chaque calque.

— Capacité de verrouiller des objets individuels (#828).
 — Déplacement de certaines propriétés globales de la carte vers le Calque de Tuiles (#149). Ça peut être utile si une carte gère des calques utilisant des tuiles de taille différente et peut-être même d'orientation différente.
 Si vous aimez au moins un seul de ces points, s'il-vous-plaît aidez-moi à y arriver plus vite en [supportant le développement de Tiled](#). Plus je recevrai de soutien, plus j'aurais les moyens de passer du temps à améliorer Tiled !

CHAPITRE 4

Modifier des Calques de Tuiles

Les tile-layer-introduction` sont ce qui fait de Tiled un *éditeur de carte par tuiles*. Même s'ils ne sont pas aussi flexibles que les :ref:`object-layer-introduction`, ils fournissent un stockage efficace des données et un bon rendu de performance ainsi qu'une création de contenu efficace. Chaque nouvelle carte en obtient une par défaut mais sentez-vous libre de la supprimer quand vous n'allez pas l'utiliser.

4.1 Brosse Tampon



Raccourci : B

L'outil le plus basique pour modifier des calques de tuiles est la Brosse Tampon. Elle peut être utilisée pour peindre aussi bien une seule tuile qu'un « tampon » plus large, c'est de là que vient son nom. En utilisant le clique droit de la souris, elle peut également capturer rapidement un tampon de tuile du calque actif. Un tampon de tuile est souvent créé en sélectionnant une ou plusieurs tuiles dans la vue Jeu de Tuiles.

La Brosse Tampon a de nombreuses fonctionnalités supplémentaires :

- En maintenant la touche Maj enfonce, cliquez à deux endroits différents pour créer deux points et dessiner une ligne entre eux.
- En maintenant les touches Ctrl+Maj enfoncées, cliquez à deux endroits différents pour créer deux points et dessiner un cercle ou une ellipse centré(e) sur le premier point.
- Activez le *Mode Aléatoire* en utilisant le bouton dé sur la barre d'outils Options d'Outils pour peindre des tuiles aléatoires du tampon de tuiles avec la Brosse Tampon. La probabilité de chaque tuile dépend autant de combien de fois elle apparaît sur le tampon de tuiles, que de la probabilité indiquée pour chaque tuile dans l'*Éditeur de Jeu de Tuiles*.
- Activez le *Mode de Remplissage de Terrain* en utilisant le bouton de tuile de terrain  dans la barre d'outils afin que la Brosse Tampon utilise des tuiles aléatoires. Ceci force les coins et contours des tuiles adjacentes à s'accorder avec celles de la tuile placée. Les tuiles de terrain sont couvertes plus en détail dans la page *Utiliser des Terrains*.
- En combinaison avec la vue *Tampon de Tuiles*, elle peut également placer aléatoirement depuis un paquet pré-défini de tampon de tuiles. Cela peut être plus utile que le *Mode Aléatoire*, qui place aléatoirement des tuiles individuelles.

- Vous pouvez inverser le tampon de tuile courant horizontalement/verticalement en appuyant sur X et Y respectivement. Vous pouvez aussi pivoter à gauche/droite en appuyant sur Z ou Maj+Z respectivement. Ces actions peuvent aussi être réalisées à partir de la barre d'outils Option d'Outil.

4.2 Brosse de Terrain



Raccourci : T

La Brosse de Terrain permet une édition efficace avec certain types de transitions de terrain (basés sur les coins, côtés ou une combinaison des deux). La mettre en place requiert d'associer les informations de terrain avec vos tuiles, ce qui est décrit en détail dans la page [Utiliser des Terrains](#).

Comme pour la *Brosse Tampon*, vous pouvez dessiner des lignes en maintenant enfoncee la touche Maj. En maintenant la touche Ctrl enfoncee, la taille de la zone d'édition est augmentée pour couvrir la tuile en entier plutôt qu'un seul coin ou côté.

En maintenant Alt, les opérations d'édition sont également appliquées avec une rotation de 180 degrés. C'est particulièrement utile lors de l'édition de cartes stratégiques où les deux côtés doivent avoir des chances équitables. Le modificateur marche bien en combinaison avec les touches Maj pour dessiner des lignes ou Ctrl pour augmenter la zone d'édition.

4.3 Outil de Seau de Remplissage



Raccourci : F

L'Outil de Seau de Remplissage fournit une manière rapide de remplir les zones vides ou les zones couvertes par les mêmes tuiles. Le tampon de tuile actuellement sélectionné sera répété dans la zone de remplissage. Il peut également être utilisé en combinaison avec le *Mode Aléatoire* ou le *Mode de Remplissage de Terrain*.

En maintenant la touche Maj enfoncee, l'outil remplit la zone sélectionnée sans prêter attention à son contenu. C'est utile pour remplir des zones personnalisées qui ont été précédemment sélectionnées en utilisant un ou plusieurs [Outils de Sélection](#).

Vous pouvez aussi inverser ou pivoter le tampon actuel comme décrit pour la *Brosse Tampon*.

4.4 Outil de Remplissage de Forme



Raccourci : P

Cet outil offre une façon rapide de remplir des rectangles ou des ellipses avec une certaine tuile ou motif. Maintenez la touche Maj pour remplir un carré ou cercle exact.

Vous pouvez aussi inverser ou pivoter le tampon actuel comme décrit pour la *Brosse Tampon*.

4.5 Gomme



Raccourci : E

Un simple outil d'effacement. Le clique gauche efface une seule tuile tandis que le clique droit est utilisé pour effacer rapidement des zones rectangulaires.

- Maintenez la touche Maj pour effacer sur tous les calques.

4.6 Outils de Sélection

Il y a de nombreux outils de sélection de tuiles qui fonctionnent de manière similaire :

- **Sélection Rectangulaire** permet la sélection de zones rectangulaires (raccourci : R)
- **Baguette Magique** permet la sélection de zones connectées remplies par la même tuile (raccourci : W)
- **Sélectionner la Même Tuile** permet la sélection des tuiles identiques à travers le calque en entier (raccourci : S)

Par défaut, chacun de ces outils remplace la zone actuellement sélectionnée. Les modificateurs suivants peuvent être utilisés pour changer ce comportement :

- Maintenir "Maj" élargit la sélection actuelle avec la nouvelle zone
- Maintenir « Ctrl » enlève la nouvelle zone de la sélection actuelle
- Maintenir « Ctrl » et « Maj » sélectionne l'intersection de la nouvelle zone avec la sélection actuelle

Vous pouvez aussi verrouiller un de ces modes (Ajout, Soustraction ou Intersection) en cliquant sur un des boutons d'outils de la barre d'outils Options d'Outil.

4.7 Gérer les Tampons de Tuile

Cela peut souvent être utile de stocker le tampon de tuile actuel quelque part pour l'utiliser à nouveau plus tard. Les raccourcis suivants œuvrent dans ce but :

- **Ctrl + 1-9** - Stocker la sélection de tuile courante. Quand aucun outil de dessin de tuile n'est sélectionné, il essaie de capturer la sélection de tuile courante (similaire à **Ctrl + C**).
- **1-9** - Sélectionne le tampon stocké à cette emplacement (similaire à **Ctrl + V**)

Les tampons de tuile peuvent aussi être stocker par nom et prolongé avec des variations en utilisant la vue *Tampons de Tuile*.

CHAPITRE 5

Travailler avec des Objets

En utilisant des objets, vous pouvez ajouter un grand nombre d'informations à votre carte à utiliser dans votre jeu. Ils peuvent remplacer des alternatives ennuyeuses telles que l'écriture en brut de coordonnées (tel que des points d'apparition) dans votre code source ou la maintenance de fichiers de données supplémentaires pour stocker des éléments de jeu.

En utilisant des *objets tuile*, des objets de types variés peuvent être facile à reconnaître ou ils peuvent être utilisés pour des raisons purement graphiques. Dans certains cas ils peuvent entièrement remplacer l'utilisation de calques de tuiles, comme démontrés dans l'exemple « Sticker Knight » fourni avec Tiled.

Tous les objets peuvent avoir des *propriétés personnalisées*, qui peuvent aussi être utilisées pour créer des connexions entre les objets.

Pour commencer à utiliser des objets, ajoutez un *Calque d'Objets* à votre carte.

5.1 Outils de Placement

Chaque type d'objet a son propre outil de placement.

Une prévisualisation de l'objet que vous êtes en train de placer est affichée lorsque vous passez votre souris sur la carte. Lors du placement d'un objet, vous pouvez appuyer sur Échap ou faire un clic droit pour annuler le placement de l'objet. Appuyez sur Échap une nouvelle fois pour sélectionner l'outil *Sélectionner des Objets*.

5.1.1 Insérer un Rectangle

Raccourci : R

Le rectangle est le premier type d'objet supporté par Tiled, voilà pourquoi les objets sont des rectangles par défaut dans le *Format de Carte TMX*. Ils sont utiles pour baliser des aires rectangulaires et pour leur assigner des propriétés personnalisées. Ils sont souvent utilisés pour spécifier des boîtes de collision.

Placez un rectangle en cliquant et glissant dans n'importe quelle direction. Maintenir Maj force un carré et maintenir Ctrl force sa taille en tant que la taille d'une tuile.

Les objets rectangles ont comme origine leur coin en haut à gauche. Cependant, si le rectangle est vide (la longueur et la hauteur sont égales à 0), il est rendu en tant qu'un petit carré autour de sa position. C'est surtout pour que l'objet soit visible et sélectionnable.

5.1.2 Insérer un Point

Raccourci : I

Les points sont les objets les plus simples que vous pouvez placer sur une carte. Ils représentent seulement un endroit, et ne peuvent être redimensionnés ou pivotés. Faites un simple clic sur la carte pour placer un objet point.

5.1.3 Insérer une Ellipse

Raccourci : C

Les ellipses marchent de la même manière que les *rectangles*, mis à part le fait qu'ils soient exportés en tant qu'ellipses. Ils sont utiles pour quand votre aire ou forme de collision a besoin de représenter un cercle ou une ellipse.

5.1.4 Insérer un Polygone

Raccourci : P

Les polygones sont le moyen le plus flexible de définir une forme pour une aire. Ils sont le plus souvent utilisés pour définir des formes de collision.

Lors du placement d'un polygone, le premier clic détermine la position de l'objet ainsi que la position du premier point du polygone. Les clics suivants sont utilisés pour ajouter des points supplémentaires au polygone. Les polygones ont besoin d'avoir au moins trois points. Cliquez sur le premier point une nouvelle fois pour finir le polygone. Vous pouvez appuyer sur Échap pour annuler la création d'un polygone.

Lorsque vous voulez changer un polygone après qu'il aie été placé, vous devez utiliser l'outil *Éditer des Polygones*.

Polylignes

Les polylignes sont créées en ne fermant pas un polygone. Faites un clic droit ou appuyez sur Entrée lors de la création d'un polygone pour le finir en tant qu'une polyligne.

Les polylignes sont rendues en tant qu'une ligne et ont besoin d'au moins deux points. Même si elles peuvent représenter des murs de collision, elles peuvent aussi être utilisées pour représenter des chemins à suivre.

Vous pouvez étendre une polyligne existante par n'importe quel bout lorsqu'elle est sélectionnée en cliquant sur les points affichés. Il est ainsi possible de finir une polyligne en la connectant à n'importe quel bout d'un autre objet polyligne. L'autre objet polyligne doit aussi être sélectionné, car les points d'interaction ne sont visibles que sur les polylignes sélectionnées.

L'outil [Éditer des Polygones](#) est aussi utilisé pour éditer des polylinéaires.

5.1.5 Insérer une Tuile

Raccourci : T

Les tuiles peuvent être insérées en tant qu'objets pour avoir une flexibilité totale dans le placement, le redimensionnement et la rotation de l'image de tuile sur votre carte. Comme tous les objets, les objets tuile peuvent aussi avoir des propriétés personnalisées qui leur sont associées. C'est utile pour le placement d'objets interactifs reconnaissables qui ont besoin d'informations spéciales, tel qu'un coffre avec un contenu défini ou un PNJ avec un script défini.

Pour placer un objet tuile, tout d'abord sélectionnez la tuile que vous voulez placer dans la vue *Jeux de Tuiles*. Ensuite, utilisez le bouton gauche de la souris sur la carte pour commencer à placer l'objet, bougez pour le placer et relâchez pour finir de placer l'objet.

Pour changer la tuile utilisée par des objets tuile existants, sélectionnez tous les objets que vous voulez changer en utilisant l'outil [Sélectionner des Objets](#) et ensuite faites un clic droit sur une tuile dans la vue *Jeux de Tuiles*, et choisissez *Remplacer la Tuile des Objets Sélectionnés*.

Vous pouvez personnaliser l'alignement des objets tuile en utilisant la propriété *Alignement d'Objet* sur le *Jeu de Tuiles*. Pour des raisons de compatibilité cette propriété est définie en tant que *Non Spécifié* par défaut, dans ce cas les objets tuiles sont alignés en bas à gauche pour toutes les orientations sauf pour les cartes *Isométriques*, où ils sont alignés en bas au centre. Définir cette propriété en tant que *En Haut à Gauche* rend l'alignement des objets tuiles consistant avec celui des *objets rectangulaires*.

5.1.6 Insérer un Modèle

Raccourci : V

Peut être utiliser pour rapidement insérer plusieurs instances d'un modèle sélectionné dans la vue Modèles. Voir [Créer des Instances de Modèle](#).

5.1.7 Insérer un Texte

Raccourci : X

Les objets textes peuvent être utilisés pour ajouter des textes avec plusieurs lignes arbitraires dans vos cartes. Vous pouvez configurer diverses propriétés de police ainsi que l'aire d'enveloppement / de coupe, ce qui les rend utile pour des notes rapides ainsi que pour du texte utilisé dans le jeu.

5.2 Sélectionner des Objets

Raccourci : S

Quand vous n'êtes pas en train d'insérer de nouveaux objets, vous utilisez généralement l'outil de Sélection d'Objet. Il comprend beaucoup de fonctionnalités qui sont mises en avant ci-dessous.

5.2.1 Sélectionner et Désélectionner

Vous pouvez sélectionner des objets en cliquant dessus ou en glissant un lasso rectangulaire, ce qui sélectionne tout objet intersectant avec cette aire. En maintenant Maj ou Ctrl tout en cliquant, vous pouvez ajouter/enlever des objets seuls dans/de la sélection. Appuyez sur Échap pour désélectionner tous les objets.

Lors de l'appui et du glissement d'un objet, cet objet sera sélectionné et déplacé. Cela vous empêche de créer une sélection rectangulaire, mais vous pouvez maintenir Maj pour forcer la sélection rectangulaire.

Par défaut vous interagissez avec l'objet le plus en haut à gauche. quand vous avez besoin de sélectionner un objet en dessous d'un autre objet, sélectionnez tout d'abord le premier objet puis maintenez Alt en cliquant au même endroit pour sélectionner des objets en-dessous. Vous pouvez aussi maintenir Alt en ouvrant le menu contextuel pour avoir une liste de tous les objets à l'endroit cliqué, pour que vous puissiez ensuite sélectionner l'objet désiré directement.

Vous pouvez passer à l'outil *Éditer des Polygones* rapidement en double-cliquant sur le polygone ou la polyligne que vous voulez éditer.

5.2.2 Déplacement

Vous pouvez simplement glisser tout objet seul, ou glisser tous les objets sélectionnés en glissant l'un d'eux. Maintenez Ctrl pour activer le suivi de grille de tuiles.

Maintenez Alt pour forcer une opération de mouvement sur les objets couramment sélectionnés, peu importe où vous cloquez sur la carte. C'est utile quand les objets sélectionnés sont petits ou couverts par d'autres objets.

Les objets sélectionnés peuvent aussi être déplacés avec les flèches directionnelles. Par défaut cela mouve les objets pixel par pixel. Maintenez Maj en utilisant les flèches directionnelles pour déplacer les objets d'une distance égale à la taille d'une tuile.

5.2.3 Redimensionnement

Vous pouvez utiliser les poignées de redimensionnement pour redimensionner un ou plusieurs objets sélectionnés. Maintenez Ctrl pour garder le même rapport hauteur/largeur de l'objet et/ou Maj pour placer l'origine du redimensionnement au centre.

Veuillez noter que vous ne pouvez changer que la longueur et la hauteur indépendamment lors du redimensionnement d'un seul objet. Lorsque plusieurs objets sont sélectionnés, le rapport hauteur/longueur est constant car il est impossible de le faire marcher pour les objets pivotés sans un support total pour les transformations.

5.2.4 Rotation

Pour pivoter, cliquez sur tout objet sélectionné pour changer les poignées de redimensionnement en poignées de rotation. Avant de tourner, vous pouvez glisser l'origine de la rotation à une autre position si nécessaire. Maintenez Maj pour pivoter par incrément de 15 degrés. Cliquez sur tout objet sélectionné une fois de plus pour revenir dans le mode redimensionnement.

Vous pouvez aussi pivoter les objets sélectionnés par incrément de 90 degrés en maintenant Z ou Maj + Z.

5.2.5 Changement de l'Ordre d'Empilage

Si le *Calque d'Objets* actif a une propriété Ordre d'Affichage mise à Manuel (De Haut en Bas par défaut), vous pouvez contrôler l'ordre d'empilage des objets sélectionnés dans le calque d'objets en utilisant les touches suivantes :

- PgHaut - Déplacer les objets sélectionnés vers le haut
- PgBas - Déplacer les objets sélectionnés vers le bas
- Menu - Déplacer les objets sélectionnés tout en haut
- Fin - Déplacer les objets sélectionnés tout en bas

Vous pouvez aussi trouver ces actions dans le menu contextuel. quand vous avez plusieurs Canques d'Objets, le menu contextuel contient aussi des actions pour déplacer les objets sélectionnés vers un autre calque.

5.2.6 Inverser des Objets

Vous pouvez inverser des objets sélectionnés horizontalement en appuyant sur X ou verticalement en appuyant sur Y. Pour les objets tuiles, cela inverse aussi leurs images.

5.3 Éditer des Polygones

Raccourci : E

Les polygones et polylinéaires ont leurs propres besoins d'édition et sont donc couverts par un outil différent, qui permet de sélectionner et de déplacer leurs nœuds. Vous pouvez sélectionner et déplacer les nœuds de plusieurs polygones à la fois. Cliquez un segment et sélectionnez les nœuds à ses deux extrémités. Appuyez sur Échap pour désélectionner tous les nœuds, ou pour revenir à l'outil *Sélectionner des Objets*.

Les nœuds peuvent être supprimés en les sélectionnant et en choisissant » Supprimer les Nœuds » depuis le menu contextuel. La touche Suppr peut aussi être utilisée pour supprimer les nœuds sélectionnés, ou les objets sélectionnés si aucun nœud est sélectionné.

Quand vous sélectionnez plusieurs nœuds à la suite dans le même polygone, vous pouvez les fusionner en choisissant « Fusionner les Nœuds » depuis le menu contextuel. Vous pouvez aussi scinder les segments entre les nœuds en choisissant « Séparer les Segments ». Alternativement, vous pouvez tout simplement double-cliquer un segment pour le séparer à l'endroit cliqué.

Vous pouvez supprimer un segment quand deux nœuds consécutifs sont sélectionnés en choisissant « Supprimer le Segment » dans le menu contextuel. Cela va convertir un polygone en polylinéaire, ou transformer un objet polylinéaire en deux objets polylinéaires.

Il est possible d'étendre une polylinéaire pour chacun de ses bouts, soit en faisant un clic droit sur ces nœuds en choisissant « Étendre la Polylinéaire », ou en passant par l'outil *Insérer un Polygone* et en cliquant sur un des bouts d'une polylinéaire déjà sélectionnée.

5.4 Connecter des Objets

Il peut souvent être utile de connecter un objet avec un autre, comme quand un interrupteur doit ouvrir une porte ou un PNJ doit suivre un chemin donné. Pour faire cela, ajoutez une propriété personnalisée de type **object** (objet) à l'objet source. Cette propriété peut être assignée à l'objet cible désiré de plusieurs façons.

Faites attention à ce que la valeur de la propriété soit sélectionnée, comme dans la capture d'écran suivante :

Ensuite, vous pouvez mettre en place la connexion en faisant d'une des actions suivantes :

- Écrire l'ID de l'objet cible.

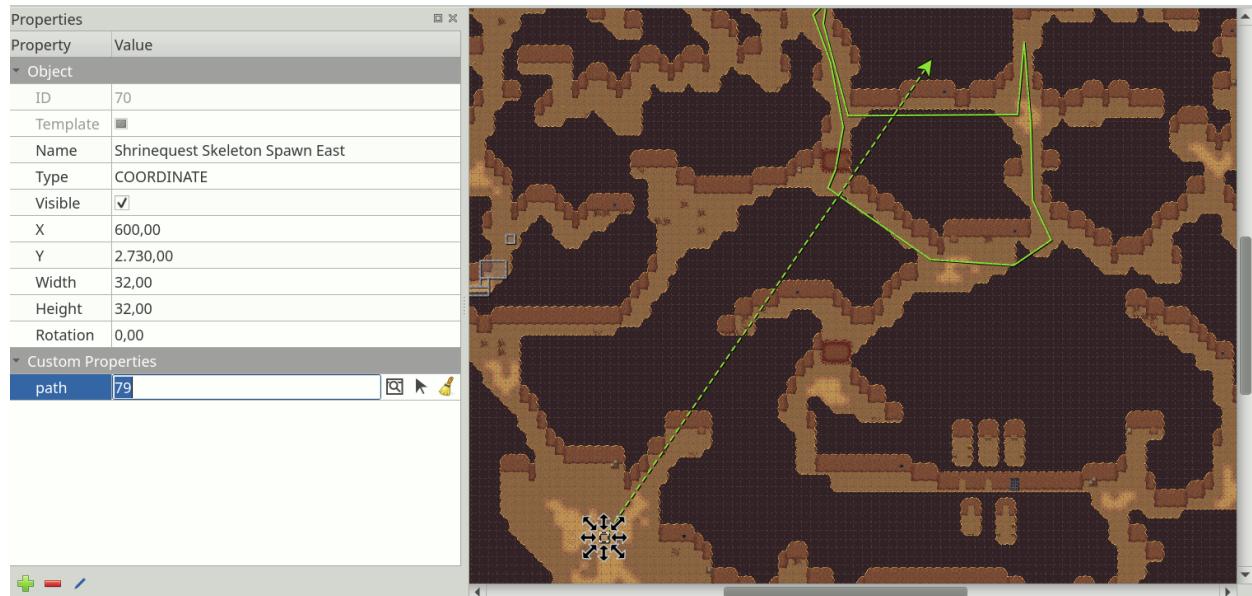


Fig. 1 – Propriété de Connexion d’Objet

- Cliquer l’icône avec la fenêtre et la loupe pour ouvrir une boîte de dialogue là où vous pouvez filtrer tous les objets de la carte pour trouver votre objet cible.
- Cliquez sur l’icône de flèche et ensuite cliquez l’objet sur la carte pour l’enregistrer en tant qu’objet cible.

Comme montré dans la capture d’écran ci-dessus, toute connexion entre objets est affichée comme des flèches qui prennent la couleur de leur objet cible (défini comme une partie des [types d’objets](#) ou par la couleur de leur calque d’objets). Vous pouvez activer/désactiver l’affichage de ces flèches en utilisant *Vue -> Montrer les Références des Objets*.

Si vous voulez vous rendre sur l’objet cible mais qu’il est très loin, vous pouvez sauter vers celui-ci en faisant un clic droit sur la propriété et en sélectionnant *Aller à l’Objet*.

Futures Extensions

Voici quelques idées d’améliorations qui peuvent être apportées aux outils ci-dessus :

- Quelques améliorations peuvent toujours être faites sur le support d’édition des polygones et polylignes, tel que la possibilité de pivoter et de redimensionner les nœuds sélectionnés ([#1487](#)).
- Les outils peuvent afficher des instructions d’usage courtes dans la barre d’état, pour aider les nouveaux utilisateurs sans avoir à les forcer à lire le manuel assidûment ([#1855](#)).

Si vous aimez au moins un seul de ces points, s'il-vous-plaît aidez-moi à y arriver plus vite en [supportant le développement de Tiled](#). Plus je recevrai de soutien, plus j’aurais les moyens de passer du temps à améliorer Tiled !

CHAPITRE 6

Modifier des Jeux de Tuiles

Pour modifier un jeu de tuiles, il doit être explicitement ouvert pour édition. Les jeux de tuiles externes peuvent être ouverts via le menu de *Fichier*, mais en général un clic sur le petit bouton *Éditer le Jeu de Tuiles* dans la barre d'outils sous le jeu de tuiles est le moyen le plus rapide d'éditer un jeu de tuiles lorsqu'il est déjà ouvert dans la vue *Jeu de Tuiles*.

6.1 Deux Types de Jeux de Tuiles

Un jeu de tuiles est une collection de tuiles. Tiled supporte deux types de jeux de tuiles pour le moment, qui est choisi lors de la création d'un nouveau jeu de tuiles :

Basé sur une Image de Jeu de Tuiles Ce jeu de tuile a une taille fixe pour toutes les tuiles ainsi qu'une images depuis laquelle les tuiles sont censées être coupées. De plus, il supporte une marge autour des tuiles ainsi qu'un espace entre les tuiles, ce qui permet d'utiliser des images de jeux de tuiles qui peuvent avoir un espace entre ou autour de ses tuiles ou ceux qui extrudent les pixels des bords de chaque tuile pour éviter un mixage de couleurs.

Collection d'Images Dans ce type de jeu de tuiles, chaque tuile référence à son propre fichier d'image. C'est utile lorsque les tuiles ne sont pas de la même taille, ou si l'assemblage des tuiles dans une texture est fait plus tard.

Peu importe le type de jeu de tuiles, vous pouvez assigner beaucoup de méta-information à lui-même ainsi qu'à ses tuiles. Certaines de ces informations peuvent être utilisées par votre jeu, tel que des *informations de collision* ou des *animations*. Les informations supplémentaires sont majoritairement dédiée à certains outils d'édition.

Note : Un jeu de tuiles peut être soit intégré à un fichier de carte, soit sauvegardé de façon externe. Depuis Tiled 1.0, l'approche recommandée et par défaut est de sauvegarder vos jeux de tuiles dans leur propre fichier. Cela permet de simplifier votre workflow car cela permet d'assurer que toute méta-information est partagée entre toutes les cartes utilisant le même jeu de tuiles.

6.2 Propriétés des Jeux de Tuiles

Vous pouvez accéder les propriétés des jeux de tuiles en utilisant l'action de menu *Jeu de Tuiles > Propriétés de Jeu de Tuiles*.

Nom Le nom du jeu de tuiles. Utilisé afin d'identifier le jeu de tuiles dans la vue *Jeux de Tuiles* lors de l'édition d'une carte.

Alignment d'Objet L'alignement à utiliser pour les *objets tuiles* font référence à des tuiles de ce jeu de tuiles. Cela affecte le placement de la tuile par rapport à la position de l'objet (l'origine) et est aussi la position autour de laquelle la rotation est appliquée.

Les valeurs possibles sont : *Non spécifié*, *En Haut à Gauche*, *En Haut*, *En Haut à Droite*, *A Gauche*, *Centre*, *A Droite*, *En Bas à Gauche*, *En Bas* et *En Bas à Droite*. Lorsque c'est non spécifié, l'alignement des objets tuiles est généralement *En Bas à Gauche*, sauf pour les cartes isométriques où c'est *En Bas*.

Décalage de Dessin Un décalage de dessin en pixels, appliqué lors du rendu de n'importe quelle tuile du jeu de tuiles (dans les couches de tuiles ou en tant qu'objets tuiles). Cela peut être utile pour aligner vos tuiles à la grille.

Couleur de Fond Une couleur de fond pour le jeu de tuiles, qui peut être changée si la couleur de fond gris foncé de base n'est pas appropriée pour vos tuiles.

Orientation Quand le jeu de tuiles contient des tuiles isométriques, vous pouvez changer ceci en *Isométrique*. Cette valeur, en tandem avec les propriétés **Longueur de Grille** et **Hauteur de Grille**, est prise en compte par les éléments affichés au dessus des tuiles. C'est utile lors de la spécification de l'*Information de Terrain*. Cela affecte aussi l'orientation utilisée par l'*Éditeur de Collision de Tuiles*.

Colonnes C'est une propriété à lecture unique pour les jeux de tuiles basée sur une image de jeu de tuiles, mais pour les jeux de tuiles de collection d'images, vous pouvez contrôler le nombre de colonnes utilisées lors de l'affichage du jeu de tuiles avec ceci.

Image Cette propriété existe seulement pour les jeux de tuiles basés sur une image de jeu de tuiles. La sélection de ce champ de valeur affiche un bouton *Éditer...*, qui vous permet de changer les paramètres permettant de couper des tuiles de l'image courante.

Bien sûr, comme avec la majorité des types de données dans Tiled, vous pouvez ajouter des *Propriétés Personnalisées* au jeu de tuiles.

6.3 Propriétés des Tuiles

ID L'identifiant de la tuile du jeu de tuiles (lecture unique)

Type Cette propriété réfère aux types personnalisés défini dans l'*Éditeur de Types d'Objets*. Visitez la section sur les *Tuiles Typées* pour plus d'information.

Longueur et Hauteur La taille de la tuile (lecture seule)

Probabilité Représente une probabilité relative que cette tuile soit choisie parmi d'autres options. Cette valeur est utilisée dans le *Mode Aléatoire* et par l'outil *Brosse de Terrain*.

Image Seulement utile pour les tuiles qui font partie d'un jeu de tuiles à collection d'images, cela affiche le fichier d'image et vous permet de le changer.

6.4 Information de Terrain

L'information de terrain peut être ajoutée à un jeu de tuiles pour activer l'utilisation de la *Brosse de Terrain*. Visitez la section sur la *définition de l'information de terrain*.

6.5 Éditeur de Collision de Tuiles

L'éditeur de collision de tuiles est disponible en cliquant sur le bouton d'Éditeur de Collision de Tuiles  sur la barre d'outils. Cela va ouvrir une vue dans laquelle vous pouvez créer et éditer des formes sur la tuile. Vous pouvez aussi associer des propriétés personnalisées avec chaque forme.

Normalement ces formes définissent les informations de collision pour une certaine image ou pour une tuile représentant la géométrie du niveau, mais bien sûr vous pouvez aussi les utiliser pour ajouter certaines zones sensibles à vos images pour par exemple des émetteurs de particules ou la source de coups de feu.

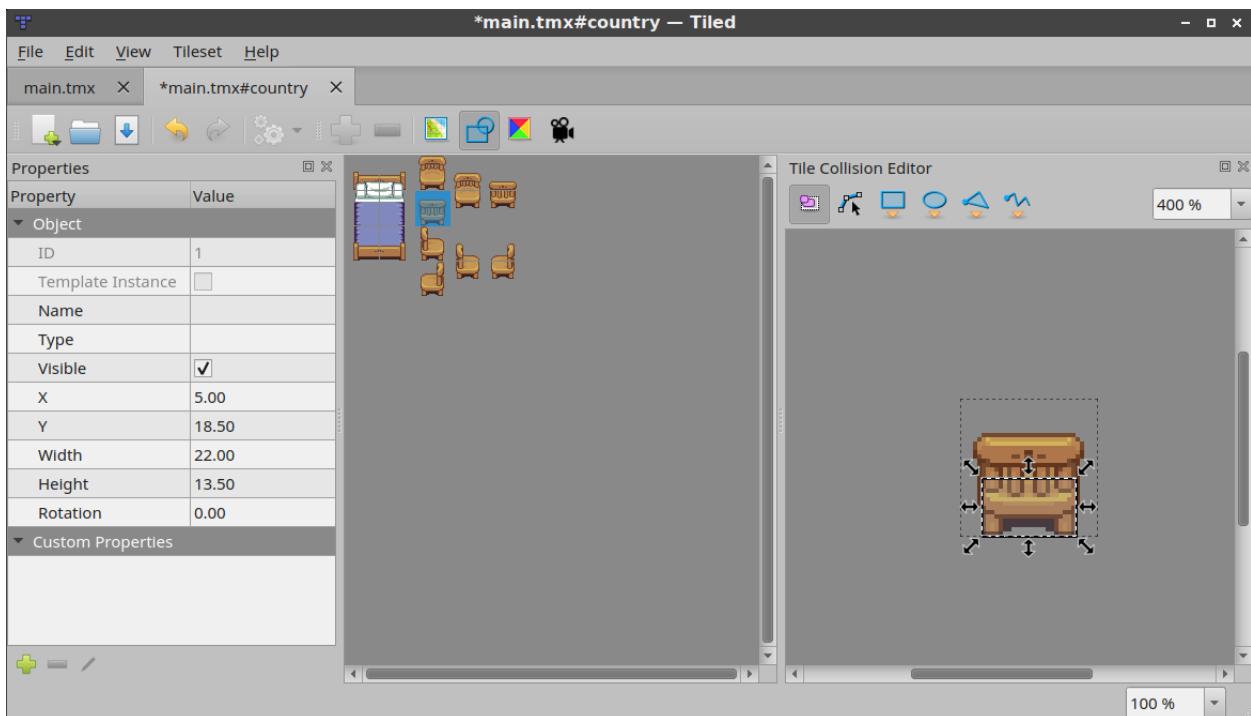


Fig. 1 – Éditeur de Collision de Tuiles

Afin de savoir rapidement si vos tuiles ont les bonnes zones de collision mises en place, elles peuvent être affichées sur la carte. Pour activer ceci, vérifier *Montrer les Formes de Collision des Tuiles* dans le menu *Vue*. Les formes de collision sont rendues pour les calques de tuiles et les objets tuiles.

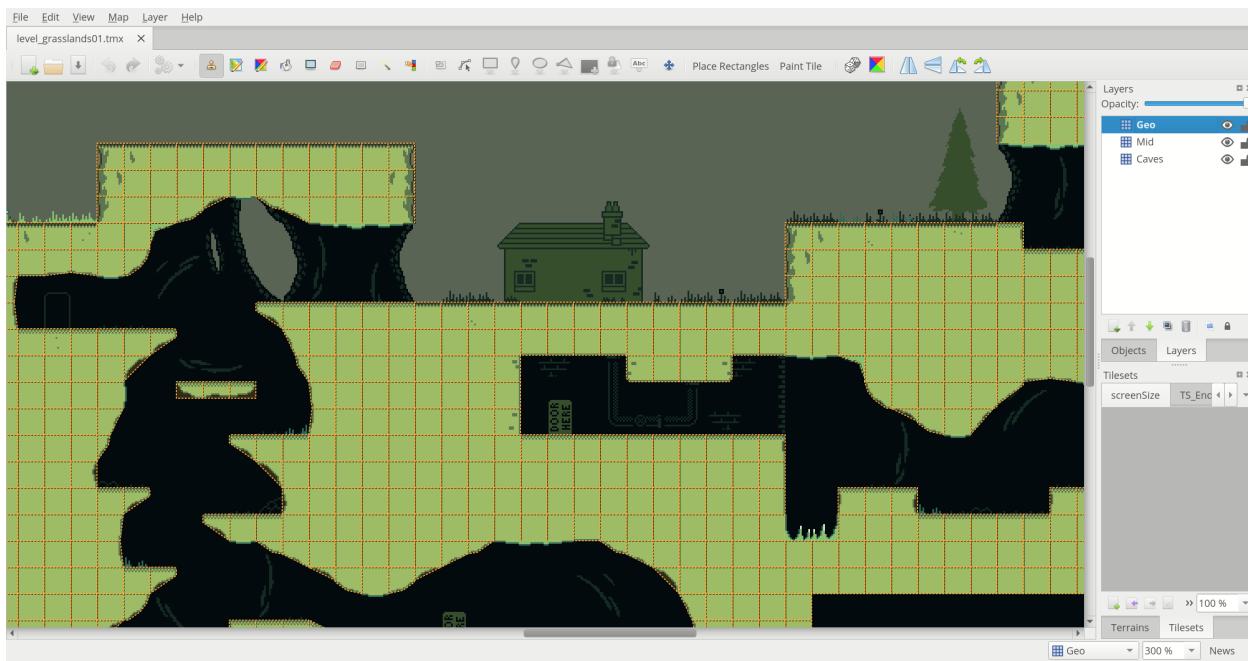


Fig. 2 – Formes de collision affichées sur la carte. Cette carte vient de *Owyn's Adventure*.

6.6 Éditeur d'Animation de Tuile

L'éditeur d'animation de tuile permet la définition d'une simple animation qui se répète pour chaque tuile en référant d'autres tuiles dans le jeu de tuiles en tant que ses trames. Ouvrez-le en cliquant sur le bouton *Éditeur d'Animation de Tuile* .

Les animations de tuiles peuvent être prévisualisées en temps réel sur Tiled, ce qui est utile pour voir de façon grossière ce à quoi elles vont ressembler en jeu. La prévisualisation peut être activée ou désactivée à travers *Vue > Montrer l'Animation des Tuiles*.

Les étapes suivantes permettent d'ajouter ou d'édition une animation de tuile :

- Sélectionnez la tuile dans la fenêtre principale de Tiled. Cela va forcer la fenêtre *Éditeur d'Animation de Tuile* à afficher l'animation (initialement vide) associée à cette tuile, ainsi que toutes les autres tuiles de ce jeu de tuiles.
- Glissez les tuiles depuis la vue du jeu de tuiles dans l'Éditeur d'Animation de Tuile das la liste à gauche pour ajouter les trames d'animation. Vous pouvez glisser plusieurs tuiles en même temps. chaque nouvelle trame a une durée par défaut de 100 ms (ou une autre valeur utilisant le champ *Durée de la Trame* en haut).
- Double-cliquez sur la durée d'une trame pour la changer.
- Glissez les trames à travers la liste pour les réordonner.

Une prévisualisation de l'animation est visible dans le coin en bas à gauche.

Vous pouvez changer la durée de plusieurs trames à la fois en les sélectionnant, en changeant la valeur dans le champ *Durée de la Trame* et enfin en cliquant sur *Appliquer*.

Futures Extensions

Il y a possiblement plein de moyens de rendre d'éditeur de jeu de tuiles plus efficace, par exemple :

Collections de Terrain

- La mise en place de tuiles de terrain est plus facile (#1729)

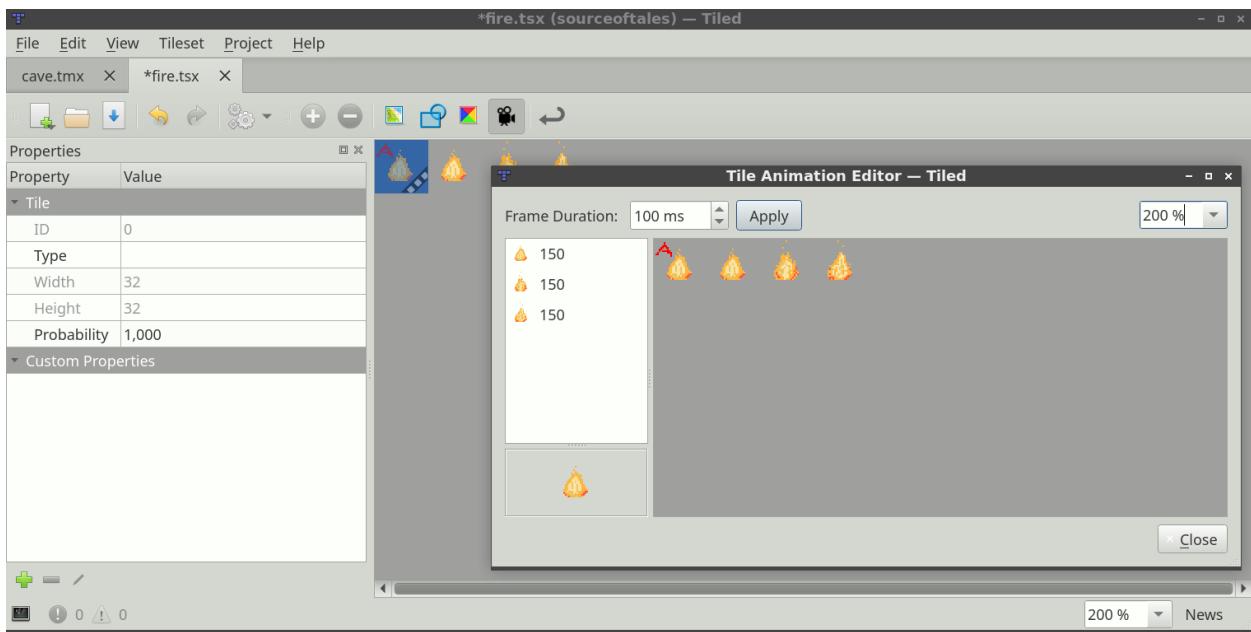


Fig. 3 – Éditeur d’Animation de Tuile

Éditeur de Collision de Tuiles

- Ajout de la possibilité d’ajouter des collisions pour plusieurs tuiles à la fois (#1322)
- Visualisation des formes de collision de tuiles dans la vue de jeu de tuiles (#1281)

Éditeur d’Animation de Tuiles

- Support de plusieurs animations nommées par tuile (#986)
- La définition d’animations affichées sur plusieurs tuiles a été rendue plus facile (#811)

Si vous aimez au moins un seul de ces points, s'il-vous-plaît aidez-moi à y arriver plus vite en [supportant le développement de Tiled](#). Plus je recevrai de soutien, plus j'aurais les moyens de passer du temps à améliorer Tiled !

CHAPITRE 7

Propriétés Personnalisées

Un des atouts de Tiled est qu'il permet de définir des propriétés personnalisées sur l'ensemble de ses structures de données. De cette manière, il est possible d'inclure de nombreuses formes d'information personnalisées, qui peuvent être ensuite utilisées par votre jeu ou par l'environnement que vous utilisez pour intégrer les cartes de Tiled.

Les propriétés personnalisées sont affichées dans la vue Propriétés. Cette vue dépend du contexte, on l'obtient généralement en affichant les propriétés du dernier objet sélectionné. Pour les tuiles dans un jeu de tuiles ou des objets sur un calque d'objets, il prend également en charge la multi-sélection.

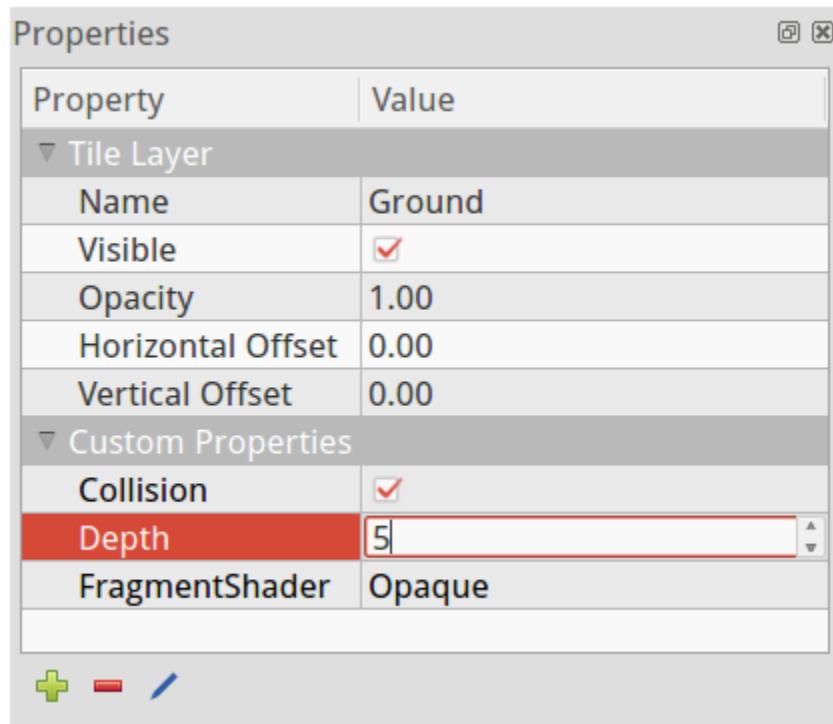


Fig. 1 – Propriétés de la Vue

7.1 Ajout de Propriétés

Lorsque vous ajoutez une propriété (en utilisant le bouton “+” en bas de la vue Propriétés), vous êtes invités à entrer son nom et son type. Actuellement Tiled prend en charge les types basiques de propriétés suivants :

- **bool** (true (vrai) ou false (faux))
- **color** (une valeur de couleur sur 32 bits)
- **file** (le chemin relatif d'un fichier)
- **float** (un nombre à virgule flottante)
- **int** (un nombre entier)
- **object** (une référence vers un objet) - *Depuis Tiled 1.4*
- **string** (n'importe quel texte, y compris un texte multi-ligne)

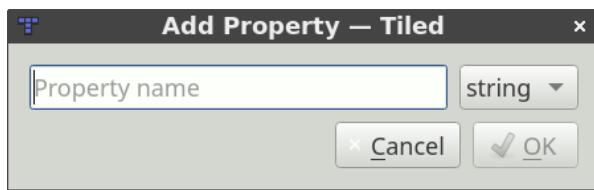


Fig. 2 – Ajouter la Boîte de Dialogue de Propriété

Le type de propriété est utilisé pour choisir un éditeur personnalisé dans la vue Propriétés. Le choix d'un type de nombre ou de booléen permet également d'éviter que la valeur obtenue en JSON et Lua soit citée dans les exportations.

Le menu de contexte pour les propriétés de fichiers personnalisées offre un façon rapide d'ouvrir le fichier dans votre éditeur préféré. Pour les références d'objet, il y a une action pour sauter rapidement vers l'objet référencé.

7.2 Héritage des Propriétés de Tuile

Lorsque des propriétés personnalisées sont ajoutées à une tuile, ces propriétés seront également visibles lorsqu'une instance de l'objet de cette tuile est sélectionnée. Cela permet de réécrire facilement les propriétés associées à une tuile pour chaque objet. Cela devient particulièrement utile lorsqu'il est combiné avec des *Tiles Typées*.

Les propriétés héritées seront affichées en gris (couleur de texte désactivée), tandis que les propriétés ré-écrites seront affichées en noir (couleur de texte habituelle).

7.3 Propriétés Prédéfinies

7.3.1 Paramètres Généraux

Normalement, vous n'utilisez qu'un nombre limité de types d'objets dans votre jeu, et chaque type d'objet a un nombre fixe de propriétés possibles, avec des types spécifiques et des valeurs par défaut. Pour économiser votre temps, Tiled permet de prédéfinir ces propriétés en se basant sur le champ « Type » des objets. Vous pouvez le mettre en place en utilisant l'Éditeur de Type d'Objet disponible depuis le menu « Vue ».

Par défaut, Tiled stocke ces types d'objets de façon globale. Cependant, comme vous voudrez souvent les partager avec les autres personnes de votre projet, vous pouvez exporter ces types d'objets ou modifier le répertoire d'enregistrement des fichiers de types d'objets *pour votre projet*. Un simple fichier XML ou JSON avec un contenu auto-explicatif est utilisé pour stocker vos types d'objets.

La couleur n'affecte pas seulement le rendu des différentes formes d'objets, mais aussi la couleur de l'étiquette qui apparaîtra si vous donnez un nom à votre objet.

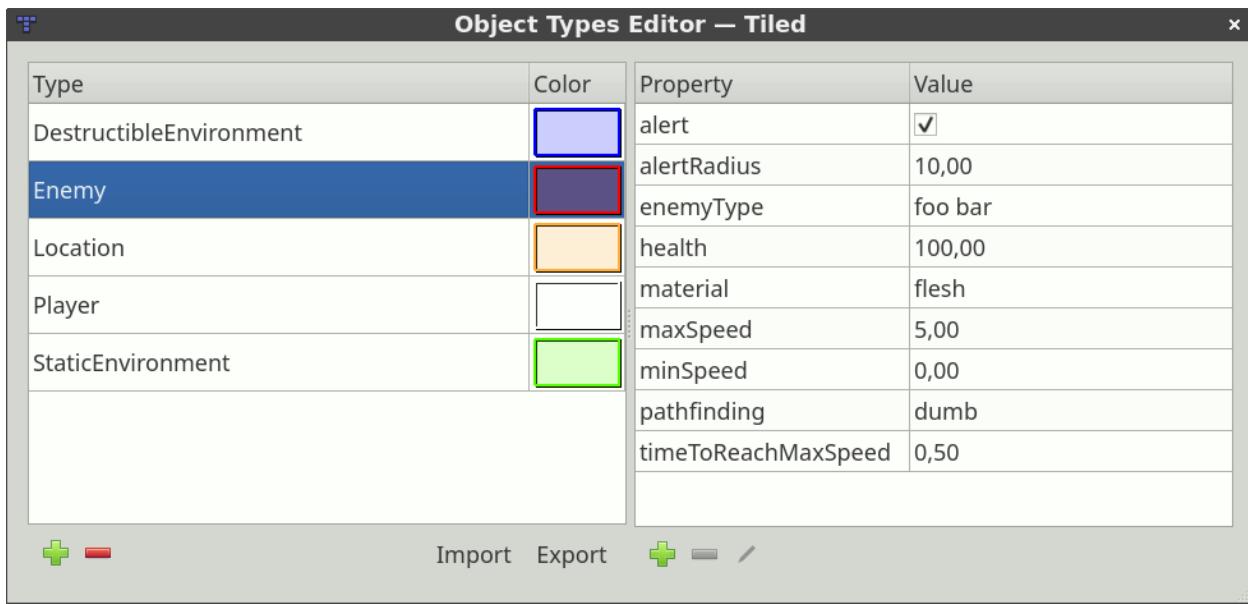


Fig. 3 – Éditeur de Type d’Objet

Pour faire apparaître les propriétés prédéfinies dans la vue Propriétés, tout ce que vous devez faire est d’entrer le nom et le type dans la propriété « Type » intégrée. Normalement, c’est ce que vous faites de toute façon pour indiquer à votre moteur de jeu quel genre d’objet il utilise.

7.3.2 Tuiles Typées

Si vous utilisez des *objets tuiles*, vous pouvez indiquer le type sur la tuile pour éviter d’avoir à l’indiquer sur chaque instance de l’objet. Mettre en place le type sur la tuile rend les propriétés prédéfinies visibles quand la tuile est sélectionnée, permettant de modifier les valeurs. Cela rend également possible l’affichage des valeurs modifiées lorsque l’on a une instance d’un objet tuile sélectionnée, permettant de les modifier une nouvelle fois.

Un exemple de cas d’utilisation serait de définir des types personnalisés comme « PNJ », « Ennemi » ou « Objet » avec des propriétés comme « Nom », « Santé » ou « Poids ». Vous pouvez ensuite spécifier les valeurs pour ces propriétés sur les tuiles qui représentent ces entités. Quand vous placez ces tuiles comme des objets, vous pouvez si besoin ré-écrire ces valeurs.

Futures Extensions

Il y a plusieurs types de propriétés personnalisées que j’aimerais ajouter :

- **Énumérations**, où vous pouvez prédéfinir toutes les valeurs possibles et que cela forme une boîte de dialogue (#1211).
- **Propriétés de liste**, qui seraient des propriétés ayant une liste de valeurs (#1493).
- **Propriétés de dictionnaire**, qui seraient des propriétés qui peuvent contenir n’importe quel nombre d’autres propriétés comme enfants (#489).

Cela serait également une bonne chose d’ajouter un support pour les **Limites des valeurs de propriétés**, comme la taille des chaînes de caractères ou un nombre minimum/maximum de valeurs.

À part pour prédéfinir les propriétés basées sur un type d’objet, J’aimerais ajouter un support pour **prédéfinir les propriétés pour chaque type de données**. Définissant ainsi que chaque propriété personnalisée est valide pour les maps, groupes de tuiles, niveaux, etc. (#1410)

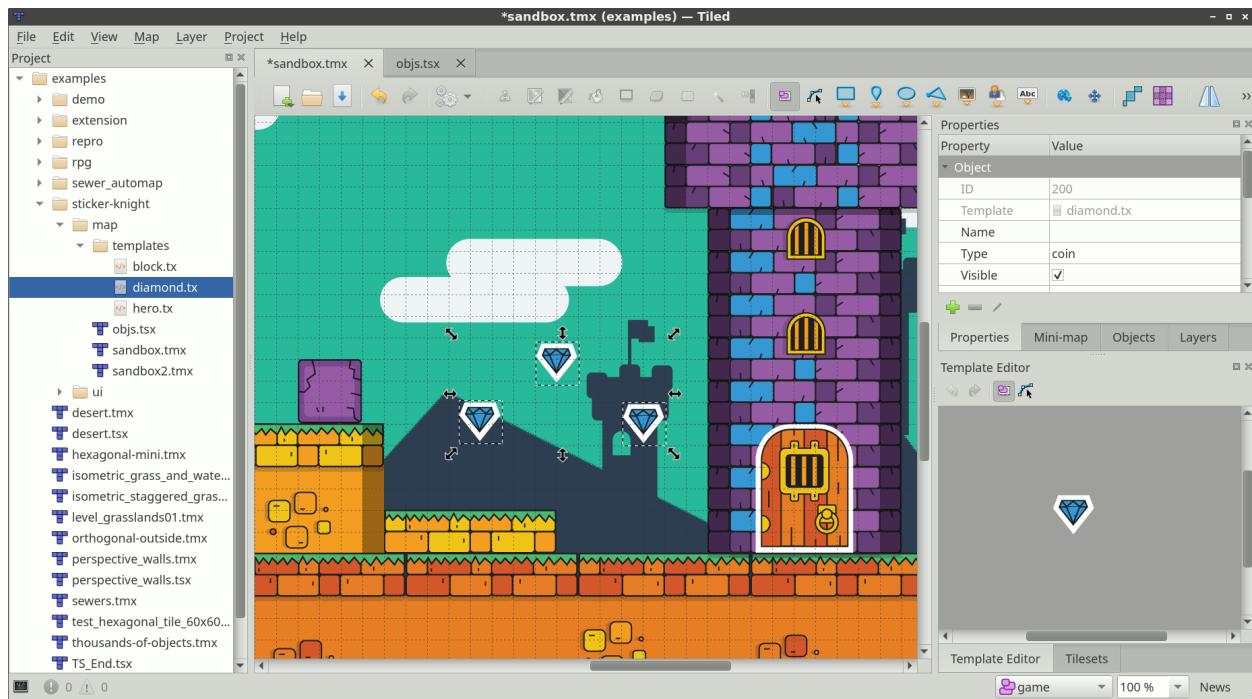
Si vous aimez au moins un seul de ces points, s'il-vous-plaît aidez-moi à y arriver plus vite en [supportant le développement de Tiled](#). Plus je recevrai de soutien, plus j'aurais les moyens de passer du temps à améliorer Tiled !

CHAPITRE 8

Utiliser des Modèles

Tout objet créé peut être sauvegardé en tant que modèle. Ces modèles peuvent ensuite être instanciés autre part en tant qu'objets qui héritent les propriétés du modèle. Cela peut vous sauver de beaucoup de travail pénible pour mettre en place le type de l'objet et ses propriétés, ou même de juste trouver la bonne tuile dans le jeu de tuiles.

Chaque modèle est stocké dans son propre fichier, qui peut être organisé dans des répertoires. Vous pouvez sauvegarder les modèles dans le format XML ou JSON, comme tout fichier de carte ou de jeu de tuiles.



8.1 Crée des Modèles

Un modèle peut être créé en faisant un clic droit sur tout objet dans la carte et en sélectionnant « Sauvegarder en tant que Modèle ». Il vous sera demandé de choisir un nom de fichier et le format dans lequel sauvegarder le modèle. Si l'objet a déjà un nom, le nom de fichier suggéré sera basé sur lui.

Pour être capable de sélectionner vos modèles pour édition ou pour les instancier, vous voudrez généralement utiliser la [vue Projet](#), donc faites attention à sauvegarder vos modèles dans un dossier qui fait partie de votre projet. Il est aussi possible de glisser un modèle depuis un explorateur de fichiers.

Note : Vous ne pouvez pas créer un modèle à partir d'un objet tuile qui utilise une tuile d'un jeu de tuiles intégré, car les [fichiers de modèle](#) ne supporte pas le référencement vers de tels jeux de tuiles.

8.2 Crée des Instances de Modèle

Raccourci : V

L'instanciation de modèle fonctionne soit en glissant-déposant le modèle depuis la vue Projet dans la carte, soit en utilisant l'outil « Insérer un Modèle » en sélectionnant un modèle et en cliquant sur la carte. La seconde option est plus pratique quand vous voulez créer plein d'instances.

8.3 Éditer des Modèles

L'édition de modèle est faite à partir de la vue *Éditeur de Modèle*. Un modèle peut être ouvert en le sélectionnant dans la vue Projet ou en glissant le fichier de modèle dans la vue *Éditeur de Modèle*. Le modèle peut aussi être sélectionné en utilisant l'action *Ouvrir un Fichier dans le Projet*.

Lors de la sélection du modèle dans la vue *Éditeur de Modèle*, la vue *Propriétés* va montrer les propriétés du modèle, là où elles peuvent être éditées.

Tout changement dans le modèle est sauvegardé automatiquement et est immédiatement appliqué à toutes les instances de ce modèle.

Si une propriété d'une instance de modèle est changée, elle va être balisée en tant que propriété modifiée en interne et ne sera pas changée quand le modèle change.

Si un fichier de modèle change sur le disque, il est automatiquement recharge et tout changement sera appliqué dans l'*Éditeur de Modèle* ainsi que dans toutes les instances de ce modèle.

8.4 Détacher des Instances de Modèle

Détacher une instance de modèle la déconnectera de son modèle, donc tout changement du modèle futur ne sera pas affecté à l'instance détachée.

Pour détacher une instance, faites un clic droit sur elle et sélectionnez *Détacher*.

Si votre chargeur de carte ne supporte pas les instances d'objet mais vous voulez toujours les utiliser, vous pouvez activer *l'option d'exportation Détacher les modèles*.

Futures Extensions

- Réinitialiser les propriétés forcées individuellement (#1725).
- Verrouiller les propriétés de modèle (#1726).
- Gestion des mauvais chemins de fichier (#1732).
- Gestion du dossier de modèles, par exemple en déplaçant, renommant ou supprimant un modèle ou un sous-dossier (#1723).

CHAPITRE 9

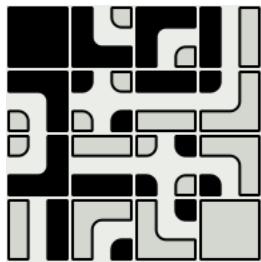
Utiliser des Terrains

Lors de l'édition d'une carte de tuiles, des fois nous ne pensons pas à des *tuiles* mais plus à des *terrains* - des aires de tuiles avec des transitions dans d'autres types de tuiles. Par exemple nous voulons dessiner un carré d'herbe, une route ou une plateforme donnée. Dans ce cas, choisir les bonnes tuiles pour les transitions différentes ou les connections peut rapidement devenir pénible. L'outil *Brosse de Terrain* a été ajouté pour rendre l'édition de carte plus facile dans ce genre de cas.

Avertissement : Même si Tiled supporte les terrains depuis la version 0.9 et a plus tard supporté une fonctionnalité similaire nommée « tuiles Wang » depuis la version 1.1, les deux fonctionnalités ont été unifiées et étendues dans Tiled 1.5. En résultat, *les informations de terrain définies dans Tiled 1.5 ne peuvent pas être utilisées dans les versions antérieures.*

La Brosse de Terrain s'appuie sur le jeu de tuiles s'il comprend une ou plusieurs *Collections de Terrain* - des collections de tuiles étiquetées par rapport à leurs agencements de terrain. Tiled supporte les collections de terrain suivantes :

Collection de Coins Les tuiles qui ont besoin de correspondre les tuiles voisines à leurs coins, avec une transition d'un type de terrain vers un autre au milieu. Une collection complète avec 2 terrains a 16 tuiles.



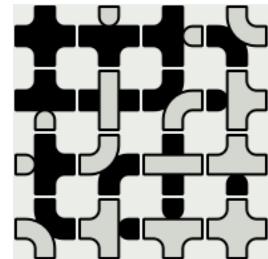
Collection de Bords Les tuiles qui ont besoin de correspondre les tuiles voisines à leurs côtés. Ceci est commun pour des routes, des clôtures ou des plateformes. Une collection complète avec 2 terrains a 16 tuiles.

Collection Mixte Les tuiles qui ont besoin de correspondre les tuiles voisines en utilisant leurs coins et leurs côtés. Cela permet de fournir plus de variations, au coût d'avoir besoin de beaucoup plus de tuiles. Une collection

complète avec 2 terrains à 256 tuiles, mais des collections réduites telles que le jeu de tuiles **Blob** de 47 tuiles peuvent aussi être utilisées avec ce type.

Basé sur les informations d'une collection de terrain, l'outil *Brosse de Terrain* peut comprendre la carte et automatiquement choisir les bonnes tuiles lors de l'édition. Si nécessaire, il peut aussi ajuster les tuiles voisines pour faire en sorte qu'elles soient connectées correctement à l'aire modifiée.

L'outil *Brosse Tampon*, ainsi que l'outil *Outil de Seau de Remplissage* et l'outil *Outil de Remplissage de Forme* ont aussi un mode dans lequel ils peuvent *remplir une aire avec un terrain aléatoire*.



9.1 Définir les Informations de Terrain

9.1.1 Créez la Collection de Terrains

Tout d'abord, passez au fichier de jeu de tuiles. Si vous regardez la carte et que le jeu de tuiles est sélectionné, vous pouvez faire ça en cliquant le petit bouton *Éditer le Jeu de Tuiles* sous la vue Jeux de Tuiles.



Fig. 1 – Le bouton Éditer le Jeu de Tuiles

Ensuite, activez le mode d'édition de terrain en cliquant sur le bouton *Collections de Terrains* sur la barre d'outils. Avec ce mode activé, les *Collection de Terrains* seront visibles, avec un bouton pour ajouter une nouvelle collection. Dans cet exemple, nous allons définir une *Collection de Coins*.

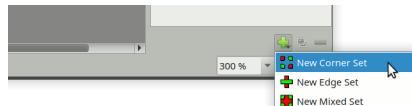


Fig. 2 – Ajouter une Collection de Terrain

Lors de l'ajout d'une collection de terrains, le nom de la nouvelle collection est automatiquement sélectionné. Donnez un nom reconnaissable à la collection, dans cet exemple nous allons écrire « Desert Ground ». Nous pouvons aussi utiliser une des tuiles comme l'icône de la collection en faisant un clic droit sur une tuile et en choisissant « Utiliser comme l'Icône de la Collection de Terrains ».

9.1.2 Ajouter des Terrains

La nouvelle collection aura un terrain ajouté par défaut. Si nous savons déjà qu'il en faudra plus, cliquez sur le bouton *Ajouter un Terrain* pour en ajouter plus.

Chaque terrain a un nom, une couleur, et peut avoir une de ses tuiles en tant qu'icône pour le rendre plus reconnaissable. Double-cliquez sur le terrain pour éditer son nom. Pour changer sa couleur, faites un clic droit sur le terrain et choisissez « Choisir une Couleur Personnalisée ». Pour assigner un icône, sélectionnez le terrain et faites un clic droit sur une tuile, puis choisissez « Utiliser comme l'Icône de la Collection de Terrains ».



Fig. 3 – Nos Terrains

Note : Nous n'aurons généralement pas besoin de définir un terrain explicitement pour les « tuiles vides ». Si vous avez des tuiles qui transitionnent vers rien, ne pas baliser ces aires devrait suffire.

Maintenant que nos terrains sont mis en place, nous sommes prêts à baliser chacune de nos tuiles.

9.1.3 Baliser les Tuiles

Veuillez noter que pour une *Collection de Coins*, nous ne pouvons baliser que les coins des tuiles. Pour une *Collection de Bords*, nous sommes limités au balisage des côtés de nos tuiles. si nous avons besoin des deux nous devons utiliser une *Collection Mixte*. Si nous découvrons que nous avons choisi le mauvais type de collection de terrains, nous pouvons toujours changer le type dans la vue Propriétés (clic droit sur la collection de terrains puis choisir *Propriétés de la Collection de Terrains...*).

Quand le terrain que nous voulons baliser est sélectionné, cliquez et glissez pour baliser les régions des tuiles qui correspondent à ce terrain.



Fig. 4 – Ici nous avons balisé tous les coins sablonneux de notre jeu de tuiles d'exemple.

Si vous faites une erreur, vous n'avez qu'à utiliser l'action Annuler (ou appuyez sur Ctrl+Z). Ou si vous remarquez une erreur plus tard, utilisez soit *Effacer le Terrain* pour enlever un type de terrain d'un coin ou sélectionnez le type de terrain correct et peignez au dessus de l'erreur. Chaque coin peut avoir un type de terrain qui lui est associé.

Maintenant faites la même chose pour chacun de vos autres types de terrain. Éventuellement vous aurez balisé toutes les tuiles sauf les objets spéciaux.

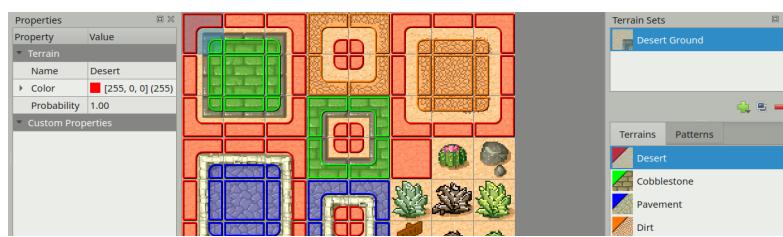


Fig. 5 – Nous avons fini de baliser les terrains de nos tuiles.

Vue de Motifs

À côté de l'onglet *Terrains* se trouve un onglet *Motifs*. Cette vue peut être utile lors du balisage de collections entières, car elle peut mettre en avant les motifs toujours manquants. Chaque motif qui apparaît déjà sur une tuile du jeu de tuiles est assombri, pour que les motifs manquants sortent du lot. Veuillez noter qu'il n'est pas nécessaire pour une collection de terrains d'avoir tous les motifs possibles, surtout si vous utilisez plus que 2 terrains.



Fig. 6 – Vue de motifs, qui montre toutes les combinaisons possibles dans la collection.

9.2 Édition avec la Brosse de Terrain

Maintenant vous pouvez désactiver le mode *Collections de Terrains* en cliquant sur le bouton dans la barre d'outils une nouvelle fois. Ensuite retournez sur la carte et activez la fenêtre *Collection de Terrains*. Sélectionnez la collection de terrains que nous venons de mettre en place, pour que nous puissions utiliser ses terrains.

Cliquez sur le terrain Sand (sable) et essayez de peindre. Vous devez tout de suite remarquer qu'il ne se passe rien. C'est parce qu'il n'y a aucune autre tuile dans la carte pour le moment, donc le terrain ne sait pas trop comment aider (car nous n'avons pas de transition vers « rien » dans notre jeu de tuiles). Il y a deux moyens de se sortir de cette situation :

- Nous pouvons maintenir Ctrl (Commande sur un Mac) pour peindre une aire un peu plus grande. De cette façon nous allons peindre au moins une seule tuile remplie avec le terrain sélectionné, cependant ce n'est pas très utile pour remplir des aires plus grandes.
- Supposons que nous allons créer une carte de désert, c'est mieux de commencer en remplissant la carte entière de sable. Vous n'avez qu'à retourner sur la fenêtre *Jeux de Tuiles* pour un moment, sélectionnez la tuile de sable et utiliser l'outil *Outil de Seau de Remplissage*.

Maintenant que nous avons du sable, sélectionnons le terrain Cobblestone (pavé). Maintenant nous pouvons voir l'outil en action !

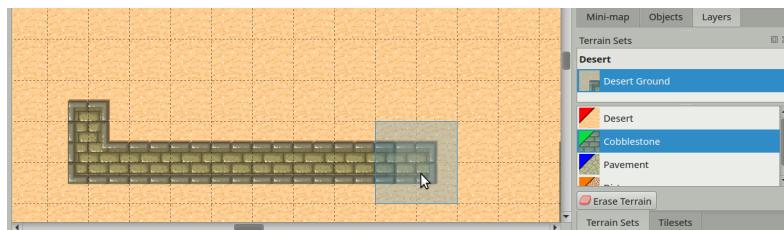


Fig. 7 – Dessiner des pavés

Enfin, regardez ce qu'il se passe quand vous essayez de dessiner de la terre sur le pavé. Comme il n'y a pas de transition de la terre directement vers le pavé, l'outil de Terrain insère tout d'abord les transitions vers le sable puis vers le pavé. Stylé !

Note : Un bouton *Effacer le Terrain* est fourni au cas où vos tuiles de terrain transitionnent vers rien. Cela permet aussi d'effacer des parties de votre terrain lors du choix des bonnes tuiles. Ce mode ne fait rien d'utille s'il n'y a pas de transition à rien dans la collection de Terrain sélectionnée.

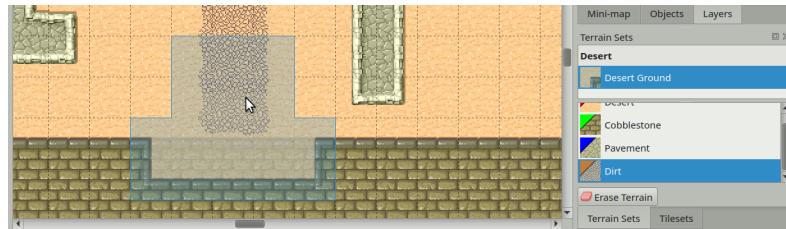


Fig. 8 – Dessiner de la terre

9.3 Mode de Remplissage de Terrain

L'outil *Brosse Tampon*, l'outil *Outil de Seau de Remplissage* et l'outil *Outil de Remplissage de Forme* ont un *Mode de Remplissage de Terrain*, qui peut être utilisé pour peindre ou remplir une aire avec des terrains aléatoires. Quand ce mode est activé, chaque cellule sera choisie aléatoirement entre toutes celles de la Collection de Terrain sélectionnée, en faisant en sorte de correspondre tous les bords et/ou coins.

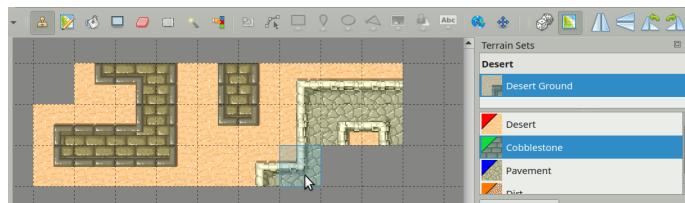


Fig. 9 – La Brosse Tampon avec le Mode de Remplissage de Terrain Activé

Veuillez noter que vu que ce mode fait en sorte que les tuiles nouvellement placées correspondent aux tuiles déjà existantes, généralement rien ne va changer lorsque vous peignez avec la Brosse Tampon sur un terrain existant. La seule exception est si vous avez plusieurs variations de la même tuile, qui dans ce cas va faire une sélection aléatoire entre celles-ci.



Fig. 10 – Le Seau de Remplissage avec le Mode de Remplissage de Terrain Activé

Lors du remplissage d'une forme ou d'une aire, seuls les bords de l'aire remplie ont besoin d'être connectés avec des tuiles existantes. Internement, l'aire est complètement aléatoire.

9.4 Probabilité de Tuile et de Terrain

Le *Mode de Remplissage de Terrain* et la Brosse de Terrain vont par défaut considérer toutes les tuiles correspondantes avec une probabilité égale. Les tuiles individuelles ainsi que les terrains ont une propriété *Probabilité*, qui peut être utilisée pour changer la fréquence à laquelle une certaine tuile ou terrain est choisie comparée aux autres options valides.

La probabilité relative d'une tuile est le produit de sa propre probabilité et de la probabilité d'un terrain pour chaque coin et/ou bord.

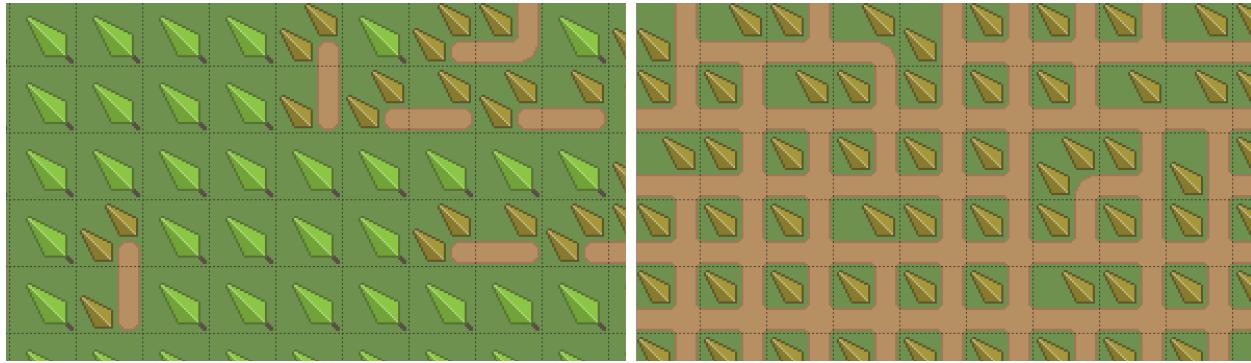


Fig. 11 – À gauche une image où « path » (chemin) a une probabilité de 0.1, et à droite une image où « path » a une probabilité de 10.

9.4.1 Probabilité des Variations

Une utilisation commune de la probabilité, surtout au niveau des tuiles individuelles, est de rendre certaines variations d'une tuile moins communes que d'autres. Notre jeu de tuiles d'exemple contient plusieurs buissons et d'autres décorations que nous voudrons aléatoirement épargiller dans le désert.

Pour ce faire, nous devons tout d'abord baliser toutes les tuiles « sand » (sable), car c'est le terrain de base. Ensuite, pour les rendre moins communes que les autres tuiles de sable, nous pouvons mettre leur probabilité à 0.01. Cette valeur veut dire qu'elles sont 100 fois plus rarement choisies que les tuiles de sables normales (qui ont toujours une probabilité de 1). Pour éditer la propriété *Probabilité* des tuiles nous devons sortir du mode *Collections de Terrains*.



Fig. 12 – Assignation d'une probabilité basse aux tuiles de décoration.

Indication : Il est aussi possible de mettre la probabilité à 0, ce qui désactive l'usage automatique d'une tuile entièrement. Cela peut être utile car les outils sont toujours conscients du terrain d'une certaine tuile, ce qui est pris en compte

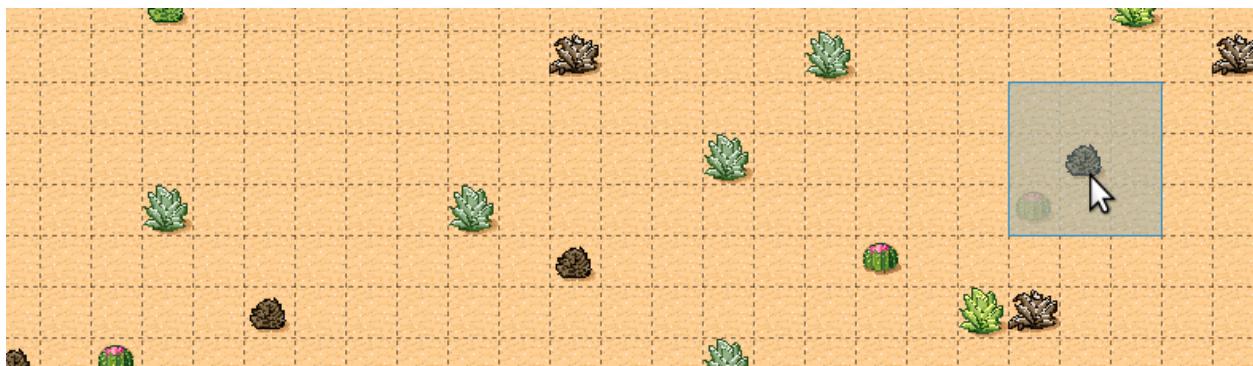


Fig. 13 – Tuiles décoratives aléatoires apparaissant avec une probabilité basse.

lors de la modification de tuiles voisines.

9.5 Transformations de Tuile

Tiled supporte les tuiles inversées et pivotées. Lors de l'utilisation des terrains, les tuiles peuvent être automatiquement inversées et/ou pivotées pour créer des variations qui ne seraient normalement pas disponibles dans un jeu de tuiles. Cela peut être activé dans les *Propriétés du Jeu de Tuiles*.

Les transformations et options liées suivants sont disponibles :

Inverser Horizontalement Autorise les tuiles à être inversées horizontalement.

Inverser Verticalement Autorise les tuiles à être inversées horizontalement. Cela doit être désactivé quand les graphismes contiennent des ombres dans une position verticale, par exemple.

Rotation Autorise la rotation des tuiles (par 90, 180 ou 270 degrés).

Préférer les Tuiles Non Transformées Quand les transformations sont activées, il peut arriver qu'un certain motif puisse être rempli soit par une tuile normale ou par une tuile transformée. Quand cette option est activée, les tuiles non transformées seront toujours prioritaires. Laisser cette option désactivée permet l'utilisation de transformations pour créer plus de variations.



Fig. 14 – Quand les rotations sont activées, le jeu de tuiles Blob qui a normalement 47 tuiles peut être réduit à seulement 15 tuiles.

9.6 Derniers Mots

Maintenant vous devez avoir une assez bonne idée de comment utiliser cet outil pour votre propre projet. Voici quelques astuces à garder en tête :

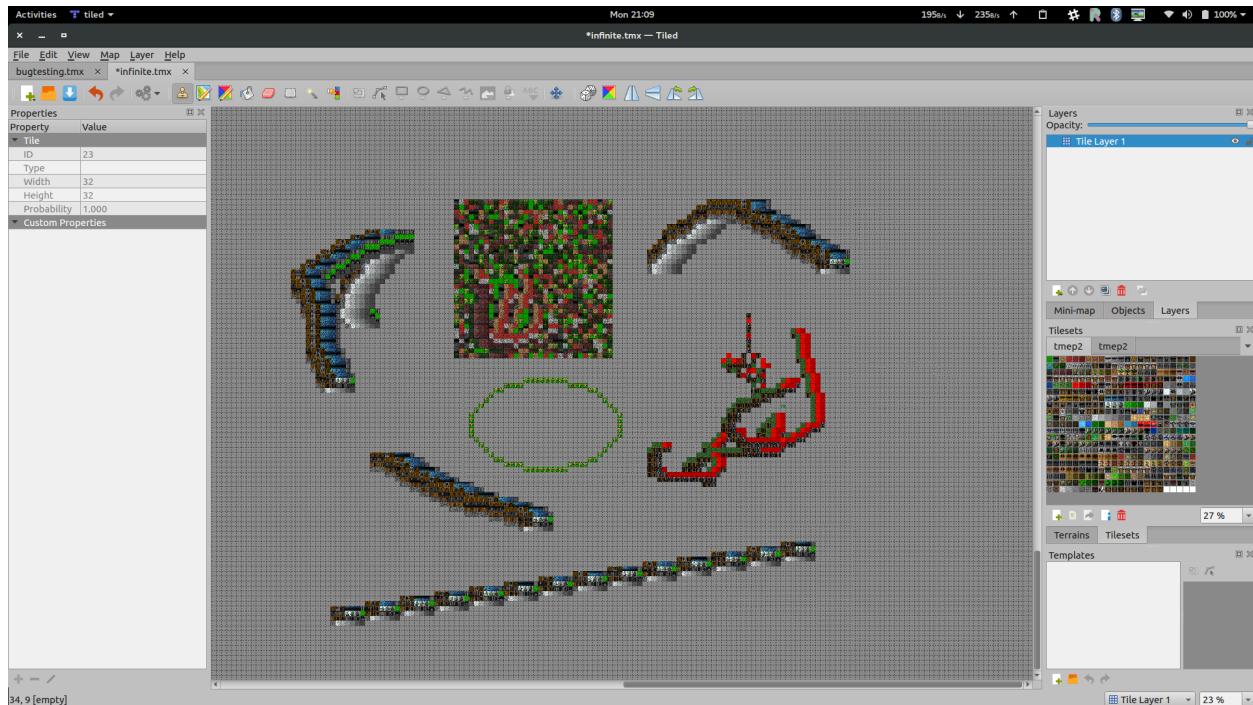
- Pour qu'un terrain interagisse avec un autre, ils doivent être dans la même *Collection de Terrains*. Cela veut aussi dire que toutes les tuiles doivent faire partie du même jeu de tuiles. Si vous avez des tuiles dans des jeux de tuiles différents avec lesquelles vous voulez créer une transition de l'une à l'autre, vous aurez besoin de fusionner les jeux de tuiles en un.
- Comme la définition des informations de terrain peut être laborieuse, vous voudrez éviter d'utiliser des jeux de tuiles intégrés pour que l'information de terrain puisse être partagée entre plusieurs cartes.

- L'outil de Terrain marche bien aussi avec les cartes isométriques. Pour faire en sorte que les informations de terrain soient affichées correctement, mettez en place l'*Orientation*, la *Largeur de la Grille* et la *Hauteur de la Grille* dans les propriétés du jeu de tuiles.
- L'outil peut gérer tout nombre de terrain (jusqu'à 255) et chaque coin de la tuile peut avoir un différent type de terrain. Cependant, il y a d'autres moyens de gérer les transitions que cet outil ne peut gérer. Aussi, il n'est pas capable d'éditer plusieurs calques à la fois. Pour un moyen plus flexible mais plus compliqué de placer des tuiles automatiquement, visitez la page sur l'["Automapping"](#).
- Il y a une [collection de jeux de tuiles](#) qui contiennent des transitions qui sont compatibles avec cet outil sur OpenGameArt.org.

CHAPITRE 10

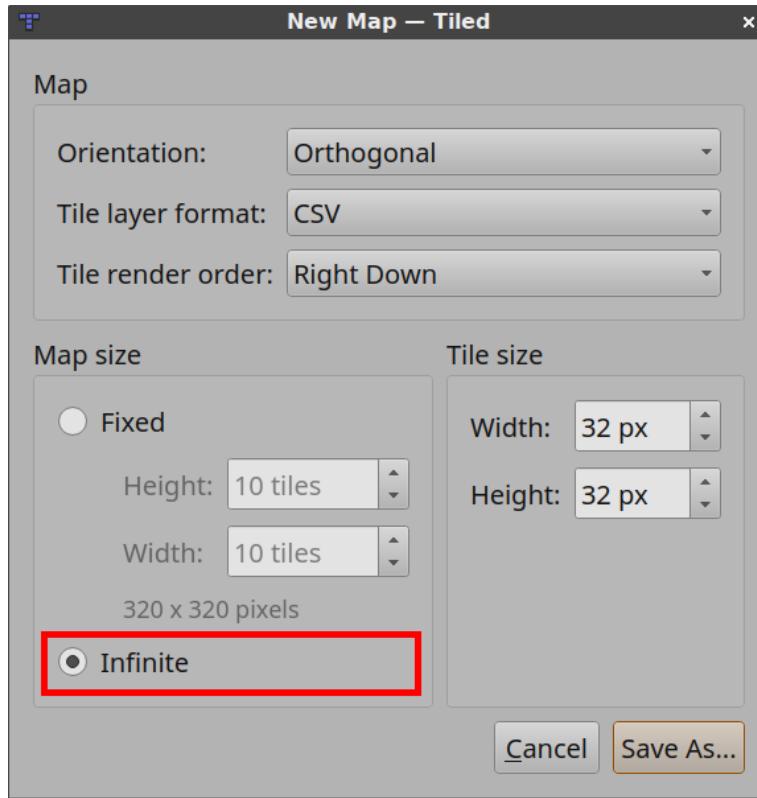
Utiliser des Cartes Infinies

Les cartes infinies vous libère des bords de la carte. La toile est « extensible automatiquement », ce qui veut plus ou moins dire que vous avez une grille infinie qui peut être peinte dans se préoccuper de la largeur ou hauteur de la carte. Les bords d'un calque particulier sont étendus quand des tuiles sont peintes en dehors des bords courants.



10.1 Créez une Carte Infinie

Pour créer une carte infinie, faites en sorte que l'option “Infinie” soit sélectionnée dans la boîte de dialogue de Nouvelle Carte.



La carte nouvellement créée aura ensuite une toile infinie.

10.2 Éditez la Carte Infinie

Mis à part l’*Outil de Seau de Remplissage* tous les outils marchent exactement de la même façon qu’avec des cartes à taille fixe. L’Outil Seau de Remplissage remplit les bords courants d’un calque de tuiles particulier. Ces bords sont agrandis lors d’un dessin supplémentaire de ce calque de tuiles.

10.3 Conversion d'une Carte Infinie vers une Carte Finie et Vice Versa

Dans les propriétés de carte, vous pouvez activer/désactiver si la carte doit être infinie ou non. Lors de la conversion d'une carte infinie vers une carte finie, la largeur et hauteur de la carte finale sont choisies basées sur les bords de tous les calques de tuiles.

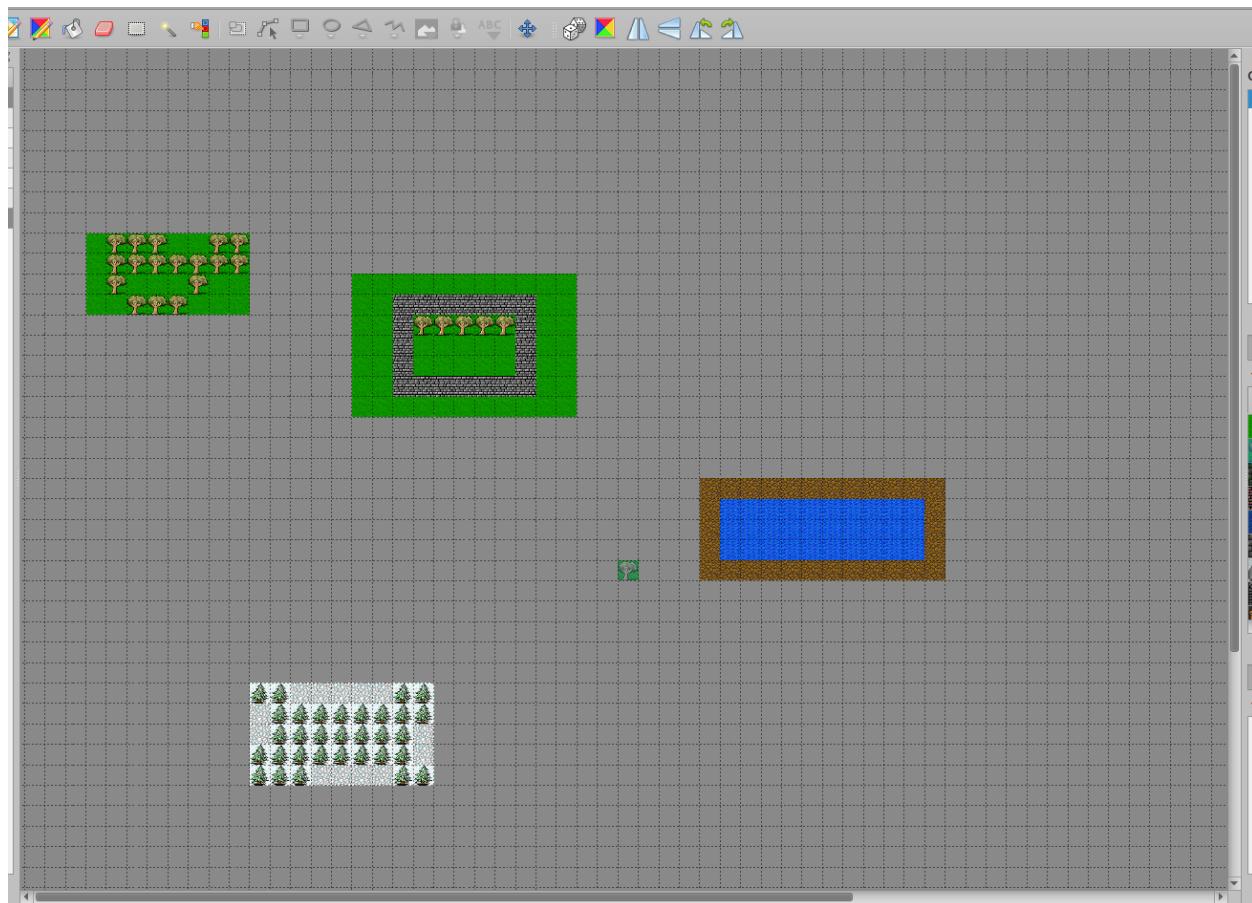


Fig. 1 – La Carte Infinie Initiale

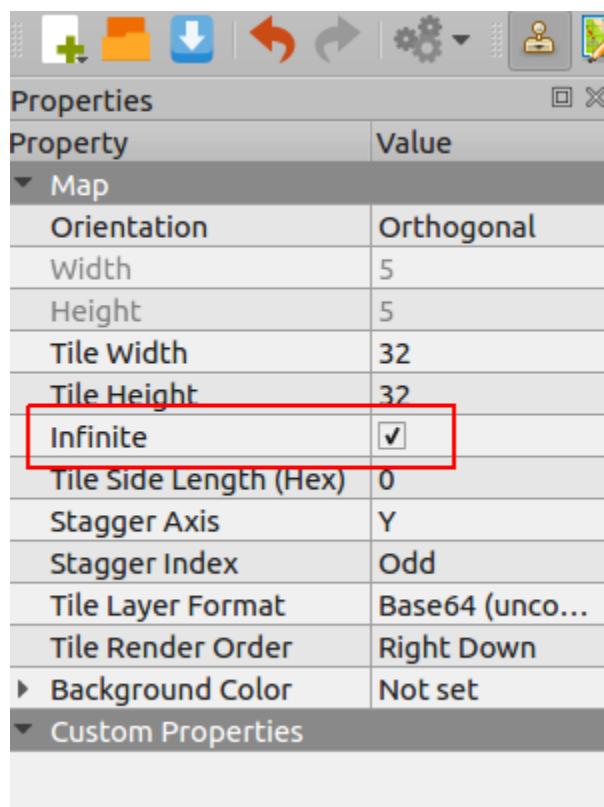


Fig. 2 – Désactiver la propriété Infinie dans les Propriétés de Carte

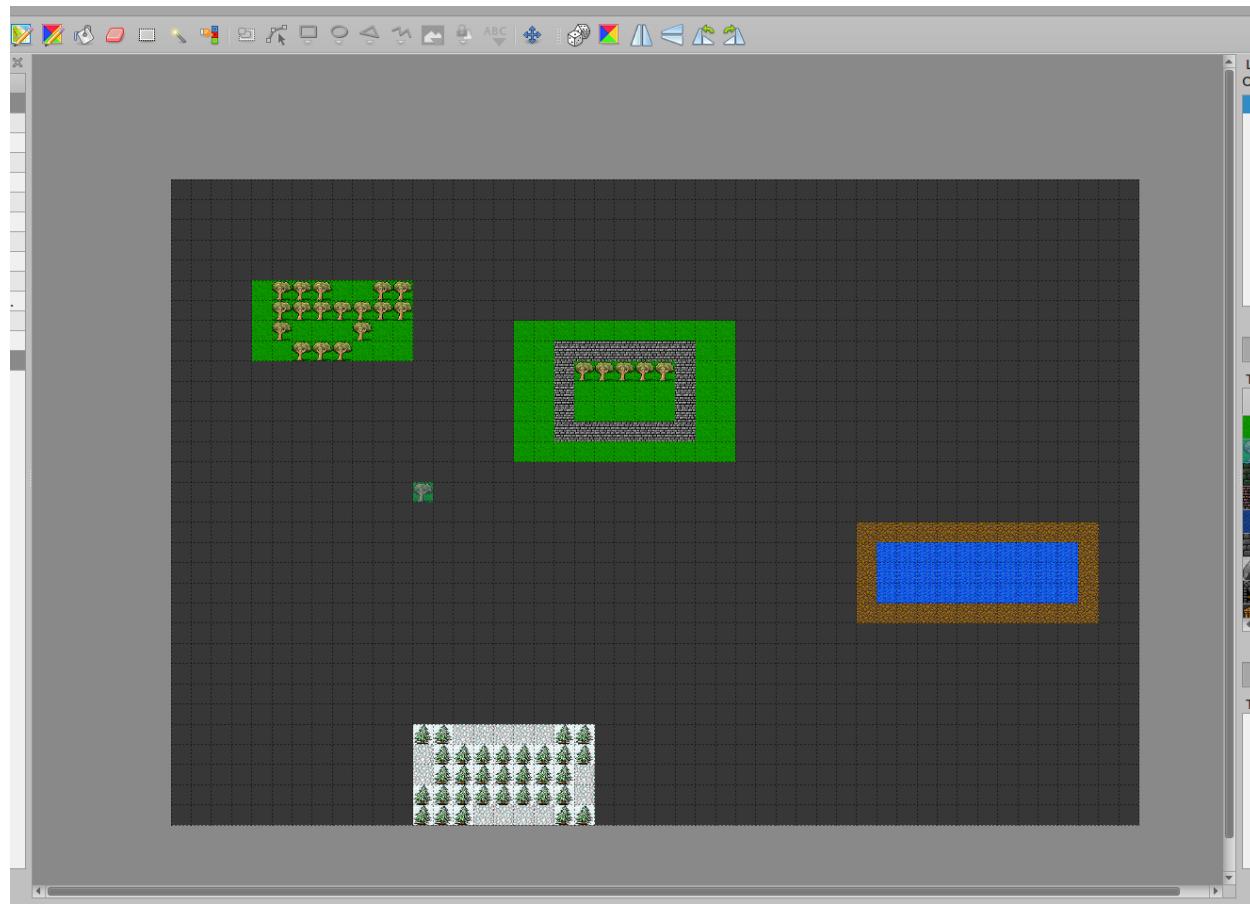


Fig. 3 – La Carte Convertie

CHAPITRE 11

Travailler avec des Mondes

Des fois, un jeu a un monde vaste qui est séparé en plusieurs cartes pour que le monde soit plus gérable par le jeu (moins d'utilisation de mémoire) ou plus facile à éditer par plusieurs personnes (éviter les conflits de fusion). Il serait utile si les cartes d'un tel monde soient visibles dans la même vue, et d'être capable de rapidement passer d'une carte à une autre pour l'édition. La définition d'un monde vous permet de faire exactement cela.

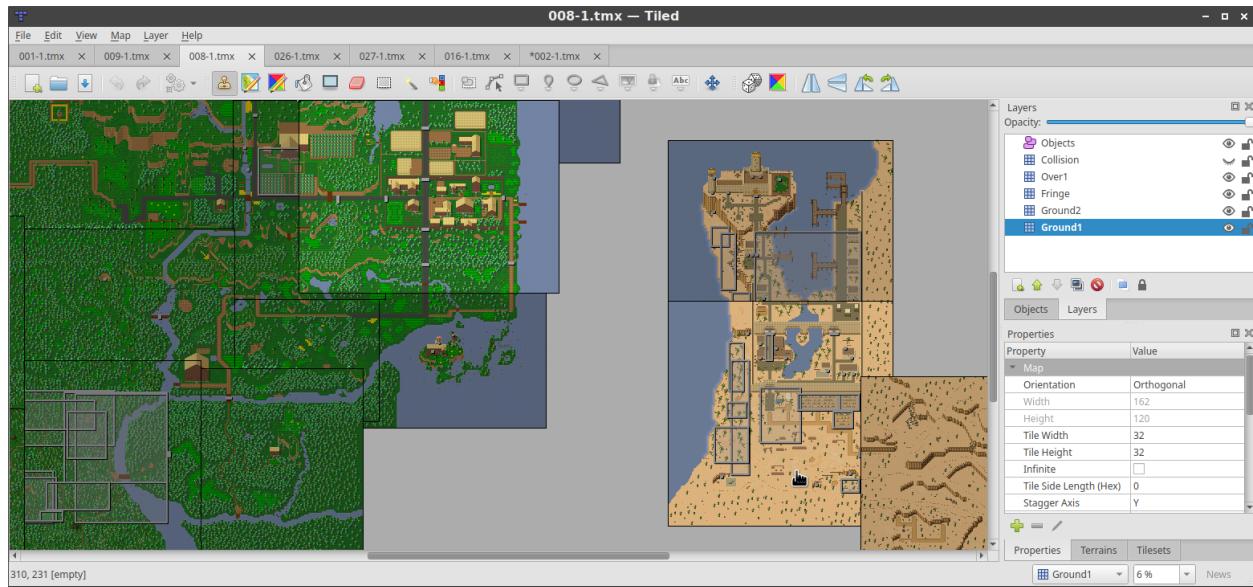


Fig. 1 – Plein de cartes de The Mana World montrées en même temps.

11.1 Définir un Monde

Un monde est défini dans un fichier `.world`, qui est un fichier JSON qui renseigne Tiled sur quelles cartes font partie du monde et à quel endroit. Les mondes peuvent être créés en utilisant l'action *Carte > Nouveau Monde...*.

Vous pouvez aussi créer des *fichiers .world* à la main. Voici un exemple simple de la définition d'un monde, qui définit la position globale (en pixels) de trois cartes :

```
{  
    "maps": [  
        {  
            "fileName": "001-1.tmx",  
            "x": 0,  
            "y": 0  
        },  
        {  
            "fileName": "002-1.tmx",  
            "x": 0,  
            "y": 3200  
        },  
        {  
            "fileName": "006-1.tmx",  
            "x": 3840,  
            "y": 4704  
        }  
    ],  
    "type": "world"  
}
```

Lorsqu'il est défini, un monde doit être chargé en choisissant *Carte > Charger un Monde...* depuis le menu. Plusieurs mondes peuvent être chargés à 1 fois, et les mondes seront automatiquement chargés une nouvelle fois lorsque Tiled sera relancé.

Quand une carte est ouverte, Tiled vérifie si elle fait partie d'un des mondes chargés. Si c'est le cas, toutes les autres cartes du même monde sont aussi chargées et affichées à côté de la carte ouverte. Vous pouvez cliquer sur n'importe quelle des autres cartes pour les ouvrir en mode édition, ce qui va changer le fichier courant tout en gardant la vue sur la même position.

Les mondes sont automatiquement rechargés quand leur fichier est changé sur le disque.

11.2 Éditer des Mondes

Lorsque vous avez chargé un monde, vous pouvez sélectionner “l’Outil Monde” depuis la barre d’outils pour ajouter, enlever ou déplacer des cartes dans le monde.

Ajouter des Cartes Cliquez sur le bouton “Ajouter la carte courante à un monde chargé” sur la barre d’outils, depuis le menu déroulant sélectionnez le monde auquel vous voulez ajouter cette carte. Pour ajouter une carte différente au monde courant, vous pouvez utiliser le bouton “Ajouter une autre carte dans le monde courant” depuis la barre d’outils. Alternativement, les deux actions peuvent être accédées en faisant un clic droit sur l’éditeur de carte.

Enlever des Cartes Appuyez sur le bouton “Supprimer la carte courante du monde courant” dans la barre d’outils. Alternativement, faites un clic droit sur l’éditeur de carte et sélectionnez l’action “Supprimer … du Monde …” depuis le menu contextuel.

Déplacer des Cartes Glissez tout simplement les cartes dans l'éditeur de carte. vous pouvez annuler le déplacement d'une carte en appuyant sur "Échap" ou en faisant un clic droit.

Alternativement vous pouvez utiliser les flèches directionnelles pour déplacer la carte couramment sélectionnée - maintenir Maj va réaliser des pas plus grands.

Sauvegarder des fichiers de Monde Vous pouvez sauvegarder des fichiers de monde modifiés en utilisant le menu *Carte > Sauvegarder le Monde*. Les mondes seront aussi automatiquement sauvegardés si vous lancez tout outil externe qui a l'option "Sauvegarder la carte avant l'exécution" activée.

11.3 Utiliser la Correspondance de Motif

Pour les projets dans lesquels les cartes suivent un certain style de nommage qui permet la définition de la position de chaque carte dans le monde en utilisant le nom de fichier, une expression régulière peut être utilisée en combinaison avec un multiplicateur et un décalage.

Note : Pour le moment aucune interface n'existe pour définir un monde en utilisant une correspondance de motif dans Tiled, et elle ne peut non plus être modifiée. Les motifs de fichiers monde doivent être édités manuellement.

Voici un exemple :

```
{
  "patterns": [
    {
      "regexp": "ow-p0*(\\d+)-n0*(\\d+)-o0000\\.tmx",
      "multiplierX": 6400,
      "multiplierY": 6400,
      "offsetX": -6400,
      "offsetY": -6400
    }
  ],
  "type": "world"
}
```

L'expression régulière est respectée pour tous les fichiers placés dans le même répertoire que le fichier de monde. Il capture deux nombres, le premier est gardé en tant que x et le deuxième en tant que y. Ceux-ci sont ensuite multipliés par `multiplierX` et `multiplierY` respectivement, et enfin `offsetX` et `offsetY` sont ajoutés. Le décalage existe surtout pour permettre plusieurs collections de cartes du même monde d'être positionnées par rapport aux autres. La valeur finale devient la position (en pixels) de chaque carte.



Fig. 2 – L'île de [Alchemic Cutie](#), en utilisant des motifs pour afficher chaque carte au bon endroit automatiquement.

Une définition de monde peut utiliser une combinaison de cartes manuellement définies et de motifs.

11.4 Seulement Montrer les Voisins Directs

Tiled fait bien attention à ne charger chaque carte, jeu de tuiles et image qu'une seule fois, mais parfois le monde est juste trop grand pour être chargé complètement. Peut-être qu'il n'y a pas assez de mémoire, ou le rendu de la carte entière est trop lent.

Dans ce cas, il y a une option pour seulement charger les voisins directs de la carte courante. Ajoutez "onlyShowAdjacentMaps": true dans l'objet JSON à la racine.

Pour que ceci soit possible, les positions mais aussi les tailles de chaque carte doivent être définies. Pour les cartes individuelles, c'est fait à travers les propriétés `width` et `height`. Pour les motifs, les propriétés sont `mapWidth` et `mapHeight`, qui sont les multiplicateurs définis par défaut par convenance. Toutes les valeurs sont en pixels.

Note : Dans le futur, une propriété peut être ajoutée pour permettre la spécification de la distance autour de la carte courante dans laquelle les autres cartes sont chargées.

CHAPITRE 12

Utiliser les Commandes

Le Bouton de Commande vous permet de créer et de lancer des commandes shell (d'autres programmes) depuis Tiled.

Vous pouvez mettre en place autant de commandes que vous le voulez. C'est utile si vous éditez des cartes pour plusieurs jeux et que vous voulez lancer des commandes pour chaque jeu. Ou vous pouvez mettre en place plusieurs commandes pour le même jeu qui charge différents points de chargement ou différentes configurations.

12.1 Le Bouton de Commande

Il est situé sur la barre d'outils principale, à droite du bouton rétablir. Cliquer dessus va lancer la commande par défaut (la première commande dans la liste de commandes). Cliquer sur la flèche à côté du bouton va afficher un menu qui vous permet de lancer toute commande que vous avez mise en place, ainsi qu'une option pour ouvrir la boîte de dialogue d'Édition de Commandes. Vous pouvez aussi trouver toutes les commandes dans le menu Fichier.

Mis à part cela, vous pouvez ajouter des raccourcis clavier personnalisés pour chaque commande.

12.2 Éditer les Commandes

La boîte de dialogue "Éditer les commandes" contient une liste de commandes. Chaque commande a plusieurs propriétés :

Nom Le nom de la commande comme il sera affiché dans la liste déroulante, pour que vous puissiez l'identifier facilement.

Exécutable L'exécutable à lancer. Il doit être soit un chemin absolu ou le nom de l'exécutable dans le PATH du système.

Arguments Les arguments pour lancer l'exécutable.

Répertoire de travail Le chemin vers le répertoire de travail.

Raccourci Une séquence de touches personnalisées pour lancer la commande. Vous pouvez utiliser "Vider" pour réinitialiser le raccourci.

Montrer la sortie dans la Console de Débogage Si cette option est activée, alors la sortie (stdout et stderr) de cette commande sera affichée dans la Console de Débogage. Vous pouvez trouver la console de Débogage dans *Vue > Vues et Barres d'Outils > Console*.

Sauvegarder la carte avant l'exécution Si cette option est activée, la carte courante sera sauvegardée avant l'exécution de la commande.

Activé Un moyen rapide de désactiver les commandes et de les enlever de la liste déroulante. La commande par défaut est la première commande activée.

Veuillez noter que si l'exécutable ou n'importe lequel de ses arguments contient des espaces, ces parties doivent être entre guillemets.

12.2.1 Variables Substituées

Dans l'exécutable, les arguments et les champs du répertoire de travail, vous pouvez utiliser les variables suivantes :

%mapfile le chemin absolu du fichier courant (soit une carte ou un jeu de tuiles).

%mappath le chemin dans lequel le fichier courant est situé.

%projectpath le chemin dans lequel le projet courant est situé.

%objecttype le type de l'objet couramment sélectionné, s'il y en a un.

%objectid l'ID de l'objet couramment sélectionné, s'il y en a un.

%layername le nom du calque couramment sélectionné.

%tileid une liste d'IDs des tuiles sélectionnés séparés par des virgules, s'il y en a.

Pour le champ de répertoire de travail, vous pouvez additionnellement utiliser la variable suivante :

%executablepath le chemin vers l'exécutable.

12.3 Commandes d'Exemple

Lancement d'un jeu personnalisé nommé « mygame » avec un paramètre -loadmap et le fichier de carte :

```
mygame -loadmap %mapfile
```

Sur Mac, rappelez-vous que les applications sont des dossiers, donc vous devez lancer l'exécutable actuel dans le dossier Contents/MacOS :

```
/Applications/TextEdit.app/Contents/MacOS/TextEdit %mapfile
```

Ou utilisez open (et veuillez noter les guillemets vu que l'un des arguments contient des espaces) :

```
open -a "/Applications/CoronaSDK/Corona Simulator.app" /Users/user/Desktop/project/main.  
lua
```

Quelques systèmes ont aussi une commande pour ouvrir des fichiers dans le programme approprié :

- OSX : open %mapfile
- Systèmes GNOME tel qu'Ubuntu : gnome-open %mapfile
- Norme FreeDesktop.org : xdg-open %mapfile

CHAPITRE 13

Automapping

13.1 Qu'est-ce que l'Automapping ?

L'Automapping est un outil avancé permettant de rechercher des combinaisons de tuiles dans les calques d'une carte et de remplacer ces parties par d'autres combinaisons de tuiles. Cela permet à l'utilisateur de dessiner des structures avec un gain de temps significatif car l'Automapping sera capable de générer des scénarios complexes, qui auraient requis bien plus de temps s'ils avaient été créés manuellement.

Le but de l'Automapping est donc que vous n'avez qu'à dessiner que dans un seul calque puis tout sera créé pour vous. Cela apporte quelques avantages :

- **Vitesse de travail** - moins de temps requis pour créer une carte.
- **Moins d'erreurs** - le but principal est de réduire la quantité d'erreurs. Si les règles ont été créées convenablement, il n'y a aucune erreur cachée.

13.1.1 Liens Externes

- L'Automapping expliqué pour Tiled 0.9 et versions ultérieures (YouTube)
- Exemples d'Automapping
- Troisième Partie du Tutoriel sur l'Éditeur de Carte Tiled : AutoMap (YouTube, en Anglais)

13.2 Mise en Place

La fonctionnalité d'Automapping cherche un fichier texte nommé "rules.txt" dans le dossier contenant la carte en cours d'édition. Chaque ligne dans ce fichier texte est soit

- un chemin vers un **fichier de règles**
- ou un chemin vers un autre fichier texte qui a la même syntaxe (ex. dans un autre dossier)
- ou un commentaire qui est indiqué par # ou //

Un **fichier de règles** est un fichier de carte standard, qui peut être lu et écrit par Tiled (*.tmx). Il peut y avoir plusieurs règles dans un fichier de règles.

Un **fichier de règles** d'automapping consiste en 4 parties majeures :

1. La définition des régions décrit quelles parties de la carte de règles seront utilisées pour créer les règles d'Automapping.
2. La définition des entrées décrit quel type de motif la carte de travail cherchera.
3. La définition des sorties décrit de quelle manière la carte de travail sera changée lorsqu'un motif d'entrée a été trouvé.
4. Les propriétés de carte sont utilisées pour affiner la localisation du motif d'entrée et la sortie de toutes les règles comprises dans ce fichier de règles.

13.2.1 Définir les Régions

Il doit y avoir soit un calque de tuiles nommé **regions**, ou il doit y avoir deux calques de tuiles nommés **regions_input** et **regions_output**. En utilisant le calque **regions**, la région définie pour l'entrée et la sortie est la même. L'utilisation de différents calques **regions_input** et **regions_output** permet d'avoir des régions différentes pour la section d'entrée et la section de sortie. Le(s) calque(s) de région est(sont) seulement utilisé(s) pour baliser des régions dans laquelle une règle d'Automapping existe. Par conséquent, peu importe les tuiles utilisées sur ce calque car ces tuiles sont utilisés pour définir une région. Vous n'avez donc qu'à placer ou ne pas placer de tuile à une coordonnée pour indiquer si cette coordonnée appartient à une règle ou non.

Si plusieurs règles sont définies sur un fichier de carte de règles, les régions ne doivent pas être adjacentes. Cela veut dire qu'il doit y avoir au moins une tuile d'espace non utilisé entre les deux règles. Si les régions sont adjacentes (cohérentes) alors les deux régions sont interprétées en tant qu'une seule règle.

Plusieurs Règles dans Un Fichier de Règles

Il est possible d'avoir plusieurs règles dans une carte de règles. Cependant, si vous voulez que les règles soient appliquées dans une séquence donnée, vous devez utiliser plusieurs **fichiers de règles** et définir la séquence dans le fichier **rules.txt**. Pour le moment, il y a aussi une certaine séquence dans un fichier de carte de règles. De façon générale, les régions avec une valeur Y petite sont exécutées en premier. S'il y a des régions ayant une même valeur de Y, alors la valeur X est prise en compte. Sur les cartes orthogonales, ce procédé d'ordonnancement est le même que pour la lecture dans la majorité des pays de l'Ouest (Gauche à droite, haut en bas). L'ordre dans une carte de règles peut être changé plus tard, lorsque Tiled sera capable d'utiliser plusieurs processus/processeurs. Donc, si vous voulez compter sur une certaine séquence, veuillez utiliser plusieurs cartes de règles différentes et ordonnez les dans le fichier rules.txt

13.2.2 Définition des Entrées

Les entrées sont généralement définies par les calques de tuiles dont le nom suit ce procédé :

input[not][index]_nom

où le **[not]** et **[index]** sont optionnels. Le nom du calque d'entrée apparaît après le premier tiret du bas. Le nom du calque d'entrée peut bien sûr contenir plus de tirets de bas.

Le **nom** détermine quel calque de la carte de travail est examinée. Par exemple, le calque *input_Sol* va examiner le calque nommé *Sol* de la carte de travail pour cette règle. *input_cas_de_test* va examiner le calque *cas_de_test* de la carte de travail pour cette règle.

Plusieurs calques ayant le même nom et index sont explicitement accepté et même voulu. Avoir plusieurs couches ayant le même nom et index permet de définir plusieurs tuiles possibles par coordonnée en tant qu'entrée.

L'index est utilisé pour créer des conditions d'entrée complètement différentes. Tous les calques ayant le même index sont pris en compte pour former une condition. Chacune de ces conditions sont validées séparément.

1. L'index ne peut pas contenir un tiret du bas.
2. L'index ne doit pas commencer par *not*.

3. L'index peut être vide.

S'il y a des tuiles sur les calques de tuiles standards, une de ces tuiles doivent être présente pour suivre la règle. Le **[not]** optionnel inverse la signification de ce calque. Donc s'il y a des calques **inputnot**, les tuiles placées sur ceux-ci ne peuvent pas exister sur la carte de travail dans la région examinée afin que la règle soit suivie. Vous pouvez combiner l'utilisation des calques input et inputnot pour créer des conditions de règles d'entrée aussi précises ou aussi confuses que vous avez besoin.

13.2.3 Définition des Sorties

Les sorties sont généralement définies par des calques dont le nom suit ce procédé :

output[index]_nom

ce qui est très similaire à la section sur les entrées. En premier le mot output doit être présent. Puis un **[index]** optionnel peut suivre. Le nom du calque cible apparaît après le premier tiret du bas. Le nom du calque cible peut bien sûr contenir plus de tirets du bas.

Tous les calques ayant le même index sont traités comme une seule sortie possible. Donc les index dans les sorties des règles sont seulement utilisés pour une sortie aléatoire.

Les index de la section de sortie n'ont rien à voir avec les index de la section d'entrée, ils sont indépendants. Ils sont utilisés en tant que caractère aléatoire dans la section de sortie. Dans la section d'entrée, ils sont utilisés pour définir plusieurs calques possibles en tant qu'entrée. Donc quand il y a plusieurs index dans une seule règle, la sortie sera choisie justement (distribué uniformément) entre tous les index. Donc un dé sera lancé et un seul index sera choisi. Tous les calques d'entrée portant cet index seront alors copiés dans la carte de travail.

Veuillez noter qu'aucun test de recouplement n'est effectué sur la sortie. Cela peut être activé en changeant la propriété de carte **NoOverlappingRules** à true.

13.2.4 Propriétés de Carte

Il y a trois différentes propriétés de carte qui peuvent être utilisées pour personnaliser le comportement des règles d'un **fichier de règles** :

DeleteTiles Cette propriété de carte est une propriété booléenne : elle peut être true ou false. Si des règles de ce fichier de règles sont appliquées à une certaine partie de votre carte, cette propriété de carte détermine si toutes les autres tuiles sont supprimées avant d'appliquer la règle. Imaginez une carte ayant plusieurs calques. Tous les calques ne sont pas complètement remplis. Dans ce cas-ci, toutes les tuiles de tous les calques doivent être effacés, comme ça seules les tuiles définies par les règles restent. Car si les tuiles ne sont pas supprimées avant ce procédé, il restera des tuiles d'avant à ces endroits, qui ne seront couverts par aucune tuile.

AutomappingRadius Cette propriété de carte est un nombre : 1, 2, 3... Elle détermine combien de tuiles autour de vos changements seront aussi testées pour refaire l'Automapping lors de l'Automapping en direct.

MatchOutsideMap Cette propriété de carte détermine si les règles peuvent trouver une correspondance même si leur région d'entrée est partiellement en dehors de la carte. Par défaut c'est **false** pour les cartes bornées et **true** pour les cartes infinies. Dans certains cas cela peut être utile de l'activer pour les cartes bornées. Les tuiles en dehors des limites de la carte sont simplement traitées comme vides sauf si **OverflowBorder** ou **WrapBorder** sont aussi true.

Tiled 1.0 et 1.1 agissaient comme si cette propriété était **true**, tandis que les versions de Tiled antérieures agissaient comme si cette propriété était **false**.

OverflowBorder Cette propriété de carte personnalise le comportement attendu par ma propriété **MatchOutside-Map**. Lorsque cette propriété est **true**, les tuiles en dehors des limites de la carte sont considérées comme des copies de la tuile dans la zone valide de la carte la plus proche, ce qui « déborde » les bords de la carte sur la région extérieure.

Quand cette propriété est `true`, elle implique **MatchOutsideMap**. Veuillez noter que cette propriété n'a aucun effet sur les cartes infinies (car il n'y a pas de notion de bord).

WrapBorder Cette propriété de carte personnalise le comportement attendu par ma propriété **MatchOutsideMap**.

Lorsque cette propriété est `true`, la carte « s'enroule » autour d'elle-même, ce qui veut dire que les tuiles sur un bord de la carte influence les régions sur le bord opposé et vice versa.

Quand cette propriété est `true`, elle implique **MatchOutsideMap**. Veuillez noter que cette propriété n'a aucun effet sur les cartes infinies (car il n'y a pas de notion de bord).

Si **WrapBorder** et **OverflowBorder** sont tous les deux `true`, **WrapBorder** est prioritaire sur **OverflowBorder**.

NoOverlappingRules Cette propriété de carte est une propriété booléenne : Une règle est pas autorisée à se recouper sur elle-même.

Ces propriétés concernent la carte en entier, ce qui veut dire qu'elles sont appliquées à toutes les règles faisant partie de la carte de règles. Si vous avez besoin de règles avec des propriétés différentes, vous pouvez utiliser plusieurs cartes de règles.

13.2.5 Propriétés des Calques

Les propriétés suivantes sont supportées sur un base par calque :

StrictEqual Cette propriété de calque est une propriété booléenne. Elle peut être ajoutée à des calques **input** et **inputnot** pour personnaliser le comportement pour les tuiles vides dans la région d'entrée.

Dans le mode « **StrictEqual** », les tuiles vides dans la région d'entrée correspondent à des tuiles vides dans le calque donné. Donc si un calque « **input** » contient une tuile vide dans sa région d'entrée, cela veut dire qu'une tuile vide est autorisée à cet endroit. Et si un calque « **inputnot** » contient une tuile vide dans sa région d'entrée, cela veut dire qu'une tuile vide est interdite à cet endroit.

13.3 Exemples

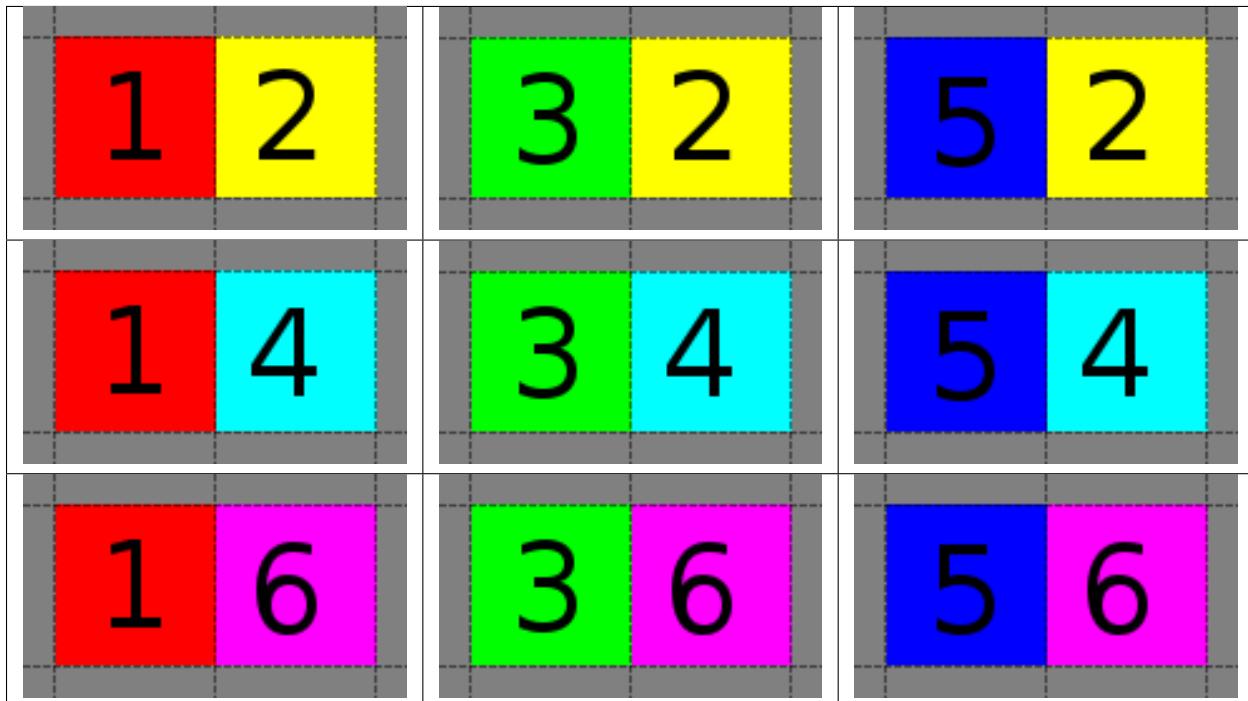
13.3.1 Exemples de Calques d'Entrée Abstraits

Avoir Plusieurs Calques d'Entrée avec le Même Nom

En sachant que les 3 calques de tuiles sont des entrées, quelles entrées possibles sont dans la carte de travail ?

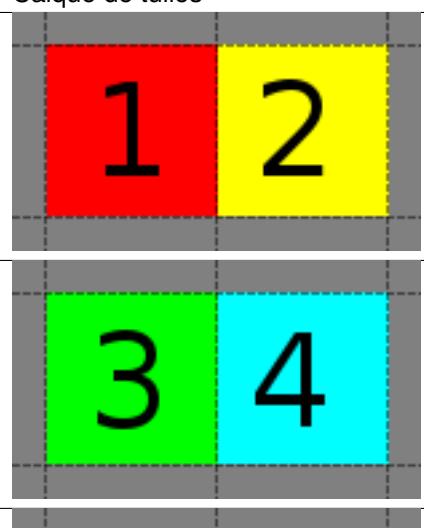
Calque de tuiles	Nom
	input_Ground
	input_Ground
	input_Ground

Les parties suivantes seraient détectées en tant que correspondance pour cette règle :

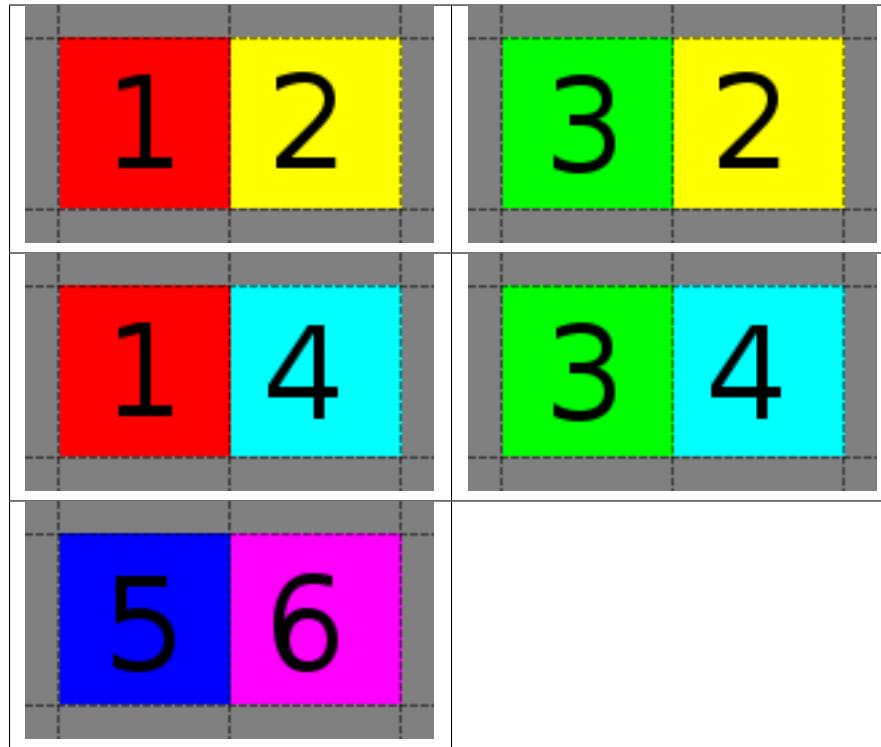


Calques d'Entrée Utilisant Plusieurs Index

Étant donné les 3 calques de tuiles suivants :

Calque de tuiles	Nom
	input_Ground
	input_Ground
	input2_Ground

Le dernier calque a un index inégal aux autres index (qui sont vides). Toutes les parties suivantes seraient reconnues en tant que correspondances dans cette carte de travail :



13.3.2 Exemples de The Mana World

Les exemples de The Mana World vont démontrer le fonctionnement de beaucoup de fonctionnalités d'Automapping. En premier un littoral sera construit, en ajoutant en premier les parties droites puis ensuite une autre règle corrigera les coins pour qu'ils suivent le jeu de tuiles courant. Après la construction du littoral, les milieux aquatiques seront balisés comme infranchissable pour le moteur de jeu. En dernier, l'herbe doit être constituée de tuiles aléatoires en utilisant 5 tuiles d'herbe différentes.

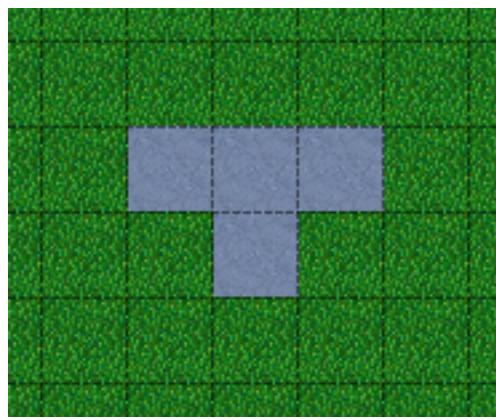


Fig. 1 – Voici ce que nous voulons dessiner.

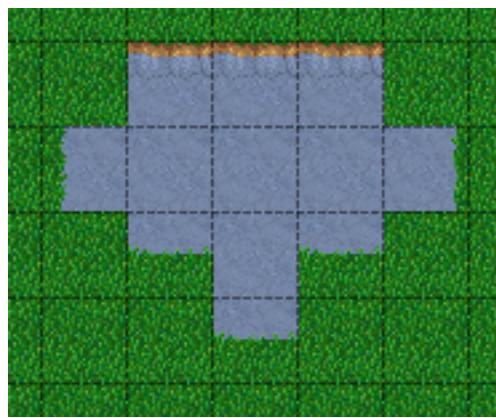


Fig. 2 – Ici nous appliquons les littoraux droits.

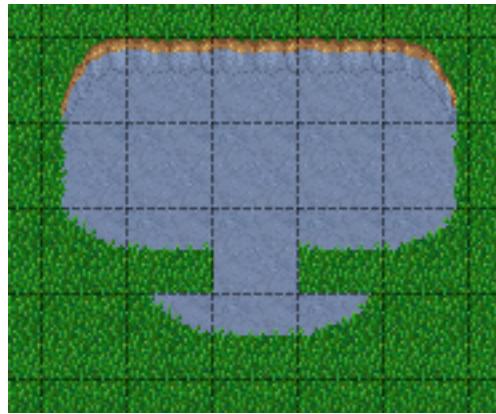


Fig. 3 – Ici nous avons quelques coins.

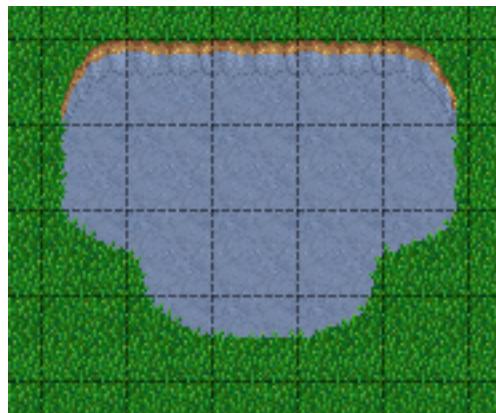


Fig. 4 – Ainsi que les coins de l'autre côté.

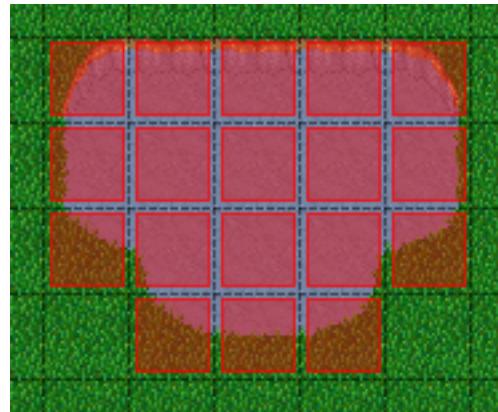


Fig. 5 – Ici toutes les tuiles infranchissables sont balisées.

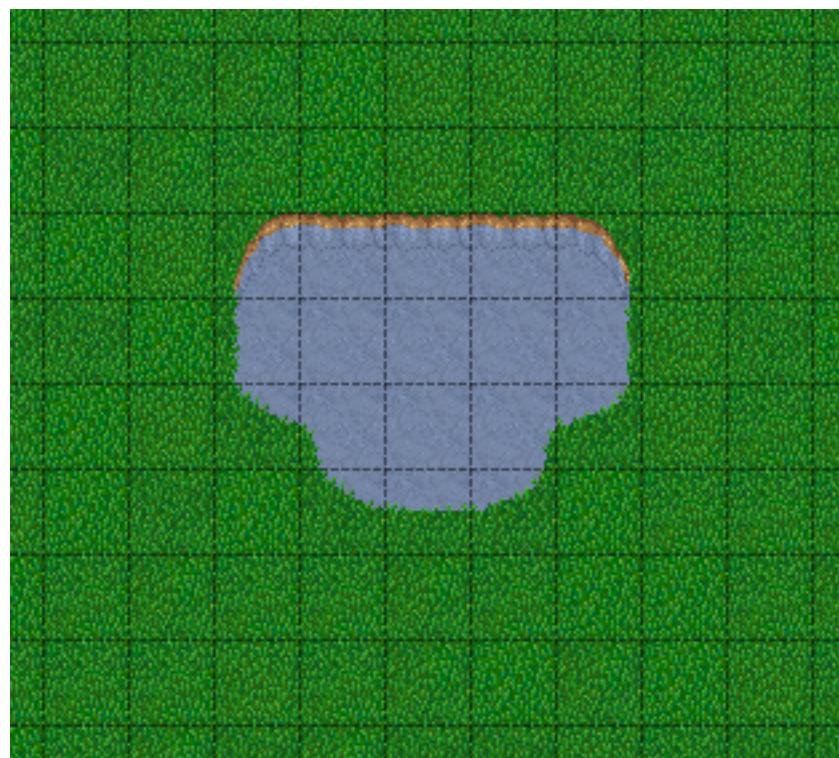
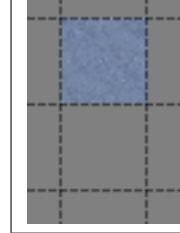
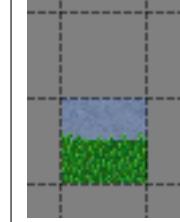


Fig. 6 – Si vous regardez l'herbe attentivement, vous verrez qu'elle est maintenant aléatoirement placée.

Littoral Basique

Cet exemple va expliquer comment un littoral droit peut être facilement mis en place entre des tuiles d'eau peu profonde et les tuiles d'herbe. Dans cet exemple, nous allons seulement ajouter le littoral, qui a de l'herbe au Sud et de l'eau au Nord.

Basiquement, la signification que nous allons définir dans la région est : *Toutes les tuiles qui sont au Sud d'une tuile d'eau et qui ne sont pas une tuile d'eau elles-même seront remplacées par une tuile de littoral*

Calque de tuiles	Nom
	regions
	input_Ground
	output_Ground

La région dans laquelle cette règle d'Automapping doit être définie est de 2 tuiles de hauteur et 1 tuile de longueur. Nous avons donc besoin d'un calque nommé *regions* et il aura 2 tuiles placées pour indiquer cette région.

Le calque d'entrée nommé *input_Ground* est représenté au milieu. Seule la tuile du haut est remplie par une tuile d'eau. La tuile du bas ne contient aucune tuile. Ce n'est pas une tuile invisible, juste aucune tuile du tout.

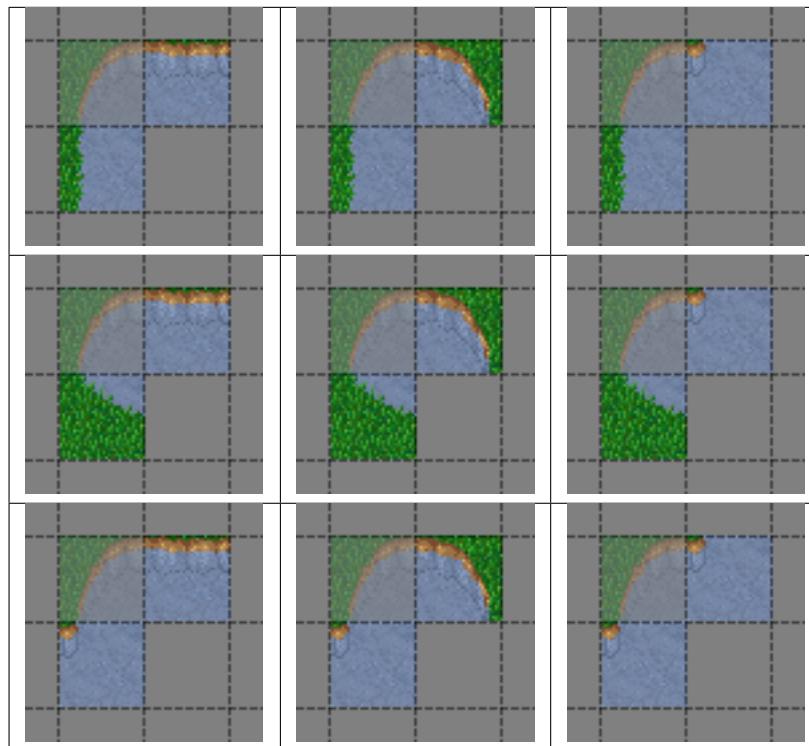
Et quand il n'y a aucune tuile dans un endroit d'une région de règle dans un calque d'entrée, quel type de tuiles va être accepté ici ? Toutes les tuiles sont acceptées sauf les tuiles utilisées dans tous les calques d'entrée ayant le même index et nom.

Ici nous avons seulement un calque de tuiles en tant que calque d'entrée ayant seulement la tuile d'eau. Donc à cette position, là où il n'y a aucune tuile, toutes les tuiles sauf cette tuile d'eau sont acceptées.

Le calque de sortie appelé *output_Ground* affiche les tuiles qui sont placées, si cette règle est suivie.

Coins pour un Littoral

Cet exemple est une continuation de l'exemple précédent. Maintenant les coins du littoral actuel doivent être implémentés automatiquement. Dans cet article nous allons examiner le coin intérieur du littoral dans le coin en haut à gauche. Les autres coins du littoral sont construits de la même façon. Donc après que l'exemple soit appliqué, nous voulons que les coins du littoral aient une tuile appropriée. Puisque nous nous basons sur le fait que l'exemple précédent a été fini, nous allons mettre les règles nécessaires pour les coins dans un autre fichier de règles. (qui est listé après dans rules.txt)



Le littoral peut avoir plus de coins proche, ce qui veut dire qu'il peut y avoir des tuiles différentes que les tuiles de littoral droites. Dans cette image nous allons voir toutes les entrées qui doivent être couvertes.

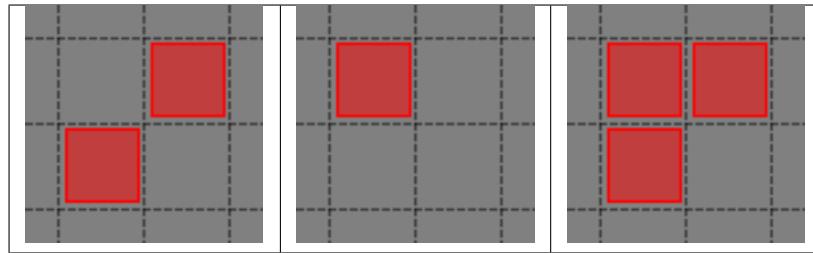
Les deux tuiles dans le coin en haut à droite et le coin en bas à gauche sont directement adjacentes à la tuile désirée (légèrement transparente) dans le coin en haut à gauche.

Nous pouvons voir 3 différentes tuiles pour le coin en bas à gauche, qui sont un littoral droit, courbé à l'intérieur ou courbé à l'extérieur.

Nous pouvons aussi voir 3 entrées différents pour le coin en haut à droite, qui sont aussi un littoral droit, courbé à l'intérieur ou à l'extérieur.

Régions d'Entrée et de Sortie

Avec cette règle, nous voulons donc placer une tuile de littoral courbé à l'intérieur dans le coin en haut à gauche, voilà pourquoi on ne prête pas attention à la tuile qui s'y trouve actuellement. Nous ne prêtons pas non plus attention à la tuile dans le coin en bas à droite. (probablement de l'eau, mais cela peut aussi être une tuile d'eau décorative, donc ignorez-la).



Nous aurons donc besoin de régions d'entrée et de sortie différentes. Dans cette image nous pouvons voir les deux calques de tuiles d'entrée et de sortie. La section d'entrée couvre juste ces deux tuiles tel que nous avons pu le voir. La région de sortie ne couvre que la seule tuile que nous voulons en sortie. Même si les régions d'entrée et de sortie ne se superposent pas, la région unie des régions d'entrée et de sortie est quand même une région cohérente, donc on a besoin d'une seule règle pour que cela marche.

Les régions de sortie peuvent être plus grandes que strictement nécessaire, car quand il n'y a aucune tuile dans la section de sortie, les tuiles dans la carte de travail ne sont pas remplacées mais sont gardées telles quelle, voilà pourquoi les régions de sortie peuvent être redimensionnées de façon à ce qu'elles soient la région unie entre les régions d'entrée et de sortie.

Calques d'Entrée

Maintenant nous voulons ajouter tous les neuf motifs possibles que nous avons pu observer comme entrée possible pour cette règle. Nous pouvons bien sûr définir neuf calques différents nommés de *input1_Ground* à *input9_Ground*. Neuf calques de tuiles ? ! Quel bazar, on va réordonner tout ça.

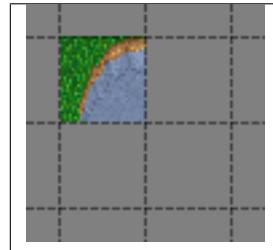
Veuillez aussi considérer ne pas avoir que 3 tuiles possibles à ces deux endroits, mais 4. Alors nous aurons besoin de $4 \times 4 = 16$ calques de tuiles pour satisfaire toutes les conditions. Un autre désavantage arrive lorsqu'on considère avoir plus de locations requises : Essayez d'avoir plus que 2 locations requises pour construire une entrée de règle. Donc pour 3 locations, alors toutes les locations peuvent avoir les 3 possibilités, donc vous aurez besoin de $3 \times 3 \times 3 = 27$ calques de tuiles. Ça ne s'arrange pas...

Essayons-donc un moyen intelligent de le faire : Tous les calques d'entrée ont le même nom, donc pour chaque position chacune des trois tuiles différentes sont valides.

Calque de tuiles	Nom
	input_Ground
	input_Ground
	input_Ground

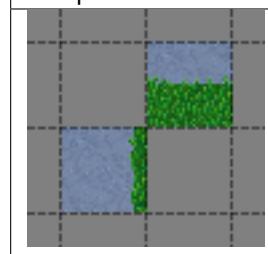
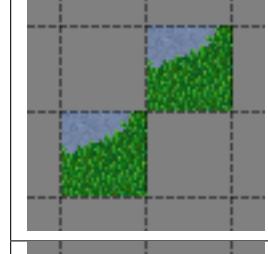
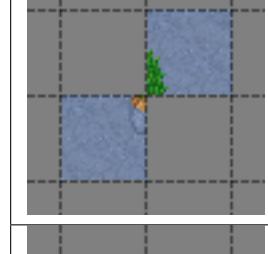
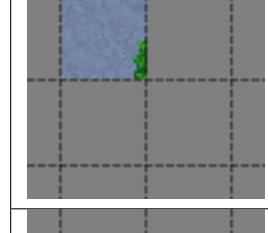
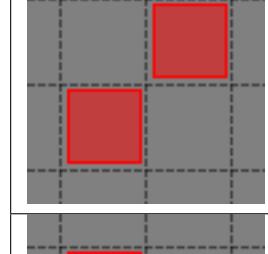
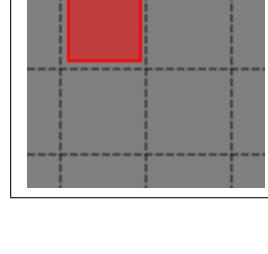
Calque de Sortie

La sortie est simple, puisqu'une seule tuile est requise. Nous avons besoin d'aucun caractère aléatoire, donc l'index ne doit pas être changé, donc il reste vide. Le calque de sortie désiré est nommé Ground, donc le nom du calque de sortie sera output_Ground. La tuile correcte est placée à l'emplacement correct dans ce calque.



Les Autres Coins d'un Littoral

Ceci est pour les coins courbés dans l'autre sens. Cet exemple utilise les mêmes concepts, juste avec des tuiles différentes.

Calque de tuiles	Nom
	input_Ground
	input_Ground
	input_Ground
	output_Ground
	regions_input
	regions_output

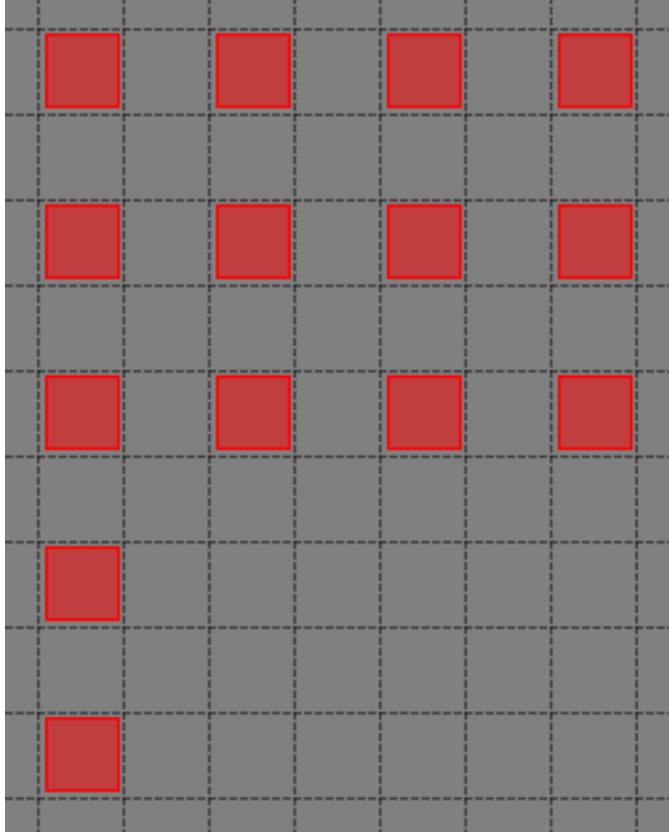
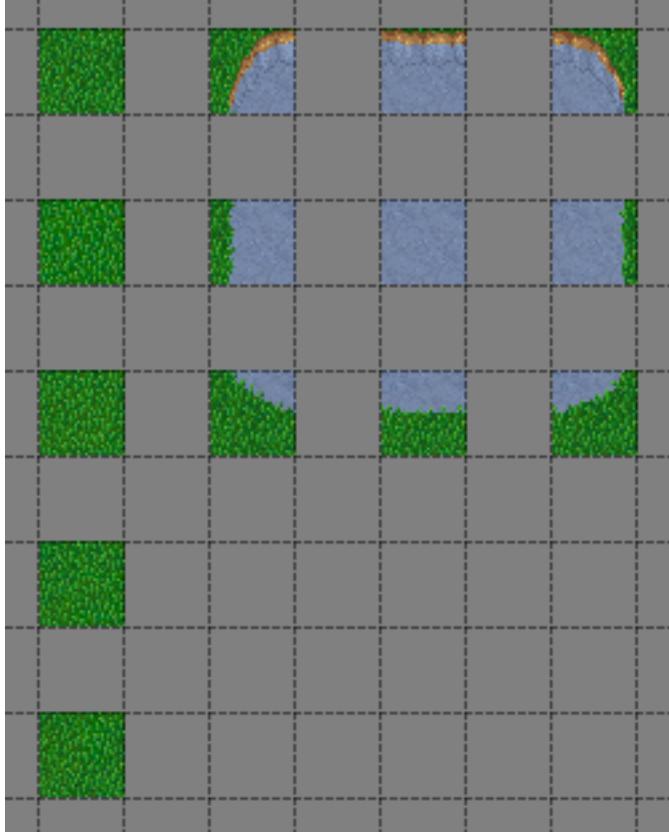
Ajouter des Tuiles de Collision

The Mana World utilise un calque de tuiles supplémentaire nommé *Collision* pour avoir des informations sur si le joueur est capable de marcher sur certaines tuiles ou non. Ce calque est invisible pour le joueur, mais le moteur de jeu l'utilise, qu'il y aie une tuile ou non.

Donc nous devons décider pour chaque position si le joueur peut y marcher et ajouter une tuile dans le calque *Collision* si elle est infranchissable.

Comme calque *d'entrée* nous allons analyser le calque *Ground* et ajouter des tuiles de collision là où le joueur ne doit pas marcher.

En réalité cette tâche est composée de beaucoup de règles, mais chaque règle en elle-même est très simple :

Calque de tuiles	Nom
	regions
	input_Ground

Dans le calque *regions* ci-dessus nous avons 14 règles différentes, car il y a 14 régions incohérentes dans le calque *regions*. Il y a 9 tuiles d'eau différentes qui doivent être infranchissables, et 5 tuiles d'herbe différentes qui seront placées aléatoirement dans le prochain exemple.

En entrée nous aurons une de toutes les tuiles utilisées et en sortie il y a une tuile dans le calque *Collision* ou non.

Avons-nous besoin de règles avec une sortie propre ? Non, ce n'est pas nécessaire si vous lancez l'Automapping une seule fois. Mais si vous concevez une carte, vous allez sans doute ajouter des aires avec collision puis enlever des parties de celle-ci et cetera.

Nous devons donc enlever les tuiles de collision de ces positions, qui ne sont plus balisées avec une collision. Cela peut être fait en ajoutant la propriété de carte *DeleteTiles* et en la changeant en *yes* ou *true*. Ainsi, toutes les parties du calque *Collision* seront effacées avant que l'Automapping soit mis en place, donc les tuiles de collision seront seulement placées sur les tuiles vraiment infranchissables et tout questionnement sur la possibilité qu'une vieille tuile de collision existe est oublié.

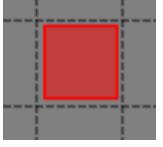
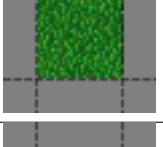
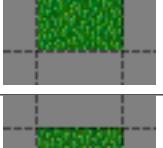
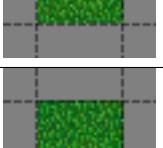
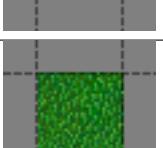
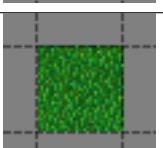
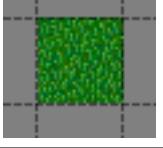
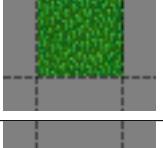
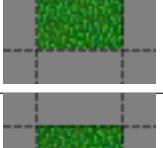
Tuiles d'Herbe Aléatoires

Dans cet exemple nous allons mélanger toutes les tuiles d'herbe, pour que toute tuile d'herbe soit chacune remplacée par une tuile choisie aléatoirement.

Nous allons prendre comme entrée toutes nos tuiles d'herbe. Cela peut être fait en ayant chaque tuile dans son propre calque d'entrée, pour que chaque tuile soit acceptée pour cette règle.

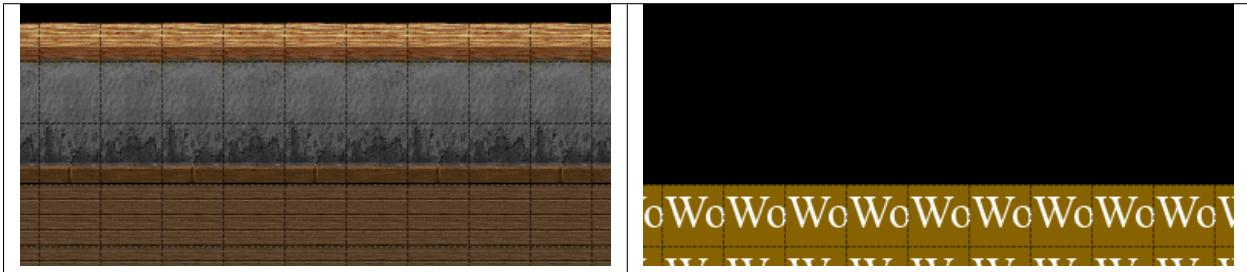
En sortie nous allons aussi placer chaque tuile d'herbe dans son propre calque de sortie. L'*index* des calques de sortie doivent être différents de tous les autres calques pour que ce processus soit aléatoire.

La règle suivante peut avoir l'air d'afficher la même chose, mais ce sont bien des tuiles d'herbe différentes. Chaque tuile d'herbe est dans un des calques d'entrée et de sortie (l'ordre des calques importe peu).

Calque de tuiles	Nom
	regions
	input_Ground
	input_Ground
	input_Ground
	input_Ground
	output1_Ground
	output2_Ground
	output3_Ground
	output4_Ground
	output5_Ground

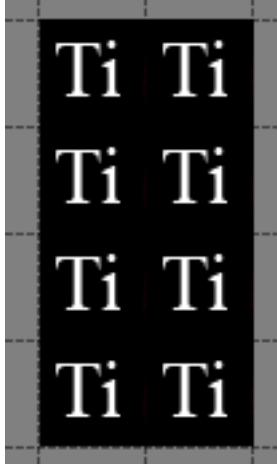
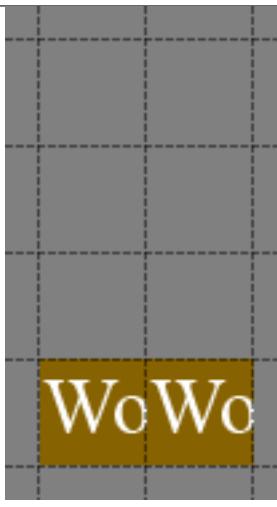
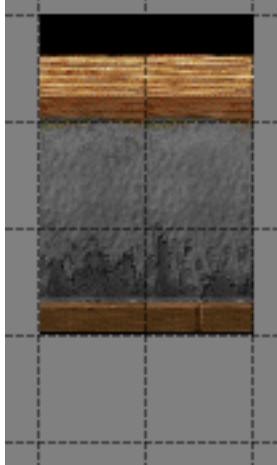
13.3.3 Un Mur Alternatif

Cet exemple va démontrer comment on peut facilement mettre en place un mur en tant que transition entre une aire explorable et le vide noir infranchissable. Un calque dédié va être utilisé en tant qu'entrée.



À mon avis un calque dédié est plus facile à utiliser pour un brouillon, mais pour ajouter des détails tel que des informations de collision sur des tuiles décoratives l'entrée doit utiliser les décorations.

La structure des calques d'entrée, de sortie et de région est très similaire à l'exemple du littoral droit de The Mana World. La différence principale est la taille différente. Comme le mur contient plusieurs tuiles de hauteur, la hauteur des calques de règles est aussi différente. Les tuiles sont aussi alternantes verticalement. Comme vous pouvez le voir sur l'image suivante, toutes les secondes tuiles qui affichent la planche à la base du mur ont une fente, par exemple.

Calque de tuiles	Nom
	regions
	input_Ground
	output_Walls

La région dans laquelle cette règle d'Automapping doit être définie doit donc être de 4 tuiles de haut et 2 tuiles de long. Nous avons donc besoin d'un calque nommé *regions* et il aura 8 tuiles placées pour indiquer cette région. Dans l'image la région du haut montrent un tel calque de région.

Le calque d'entrée a la signification suivante :

S'il y a 2 tuiles marron adjacentes horizontalement dans le calque donné et dans les 2x3 tuiles au-dessus de celles-ci il n'y a pas de tuile marron, cette règle est respectée.

Seules les 2 coordonnées les plus basses contiennent les tuiles marron. Les coordonnées du haut ne contiennent aucune tuile. (Pas de tuile invisible, juste aucune tuile du tout.) Le calque d'entrée nommé *input_set* est représenté au milieu de l'image.

La sortie consiste aussi d'un seul calque nommé *output_Walls*. Il contient les vraies tuiles du mur.

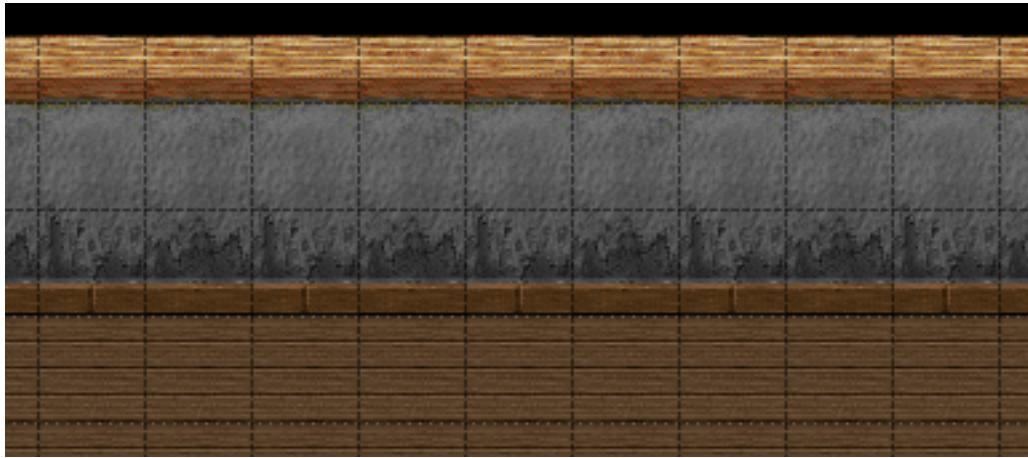


Fig. 7 – Les tuiles sont alternantes verticalement.

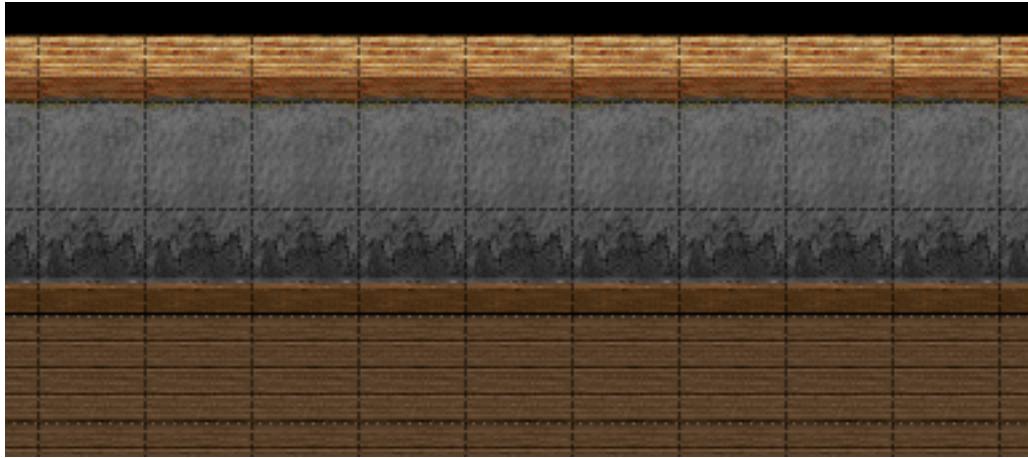


Fig. 8 – Une version défectueuse de la règle, car *NoOverlappingRules* n'a pas encore été ajouté.

Lorsqu'on essaye de correspondre le calque d'entrée au calque désiré (image de droite de l'image tout au début de l'exemple), vous allez voir qu'il correspond jusqu'au bout, peu importe l'ajustement vertical.

Voilà pourquoi quand nous utilisons cette règle telle qu'on vient de le voir, nous n'aurons pas le résultat attendu, parce que la règle se superpose à elle-même. Le problème de superposition est montré dans l'image ci-dessus.

Comme la superposition est pas désirée, nous pouvons la désactiver en ajoutant la propriété de carte *NoOverlappingRules* à la carte de règles et la mettre à *true*.

Veuillez garder en tête que la propriété de carte est appliquée pour toutes les règles dans cette carte de règles.

CHAPITRE 14

Exporter des Formats

Même s'il y a de nombreuses *bibliothèques et environnements* qui travaillent directement avec des cartes Tiled, Tiled supporte aussi un nombre de fichiers et de formats d'exportation supplémentaires, ainsi que *l'exportation de la carte vers une image*.

L'exportation peut être effectuée en cliquant sur *Fichier > Exporter*. Lorsque l'action de menu est lancée plusieurs fois, Tiled ne demandera le nom du fichier que la première fois. L'exportation peut aussi être automatisée en utilisant les paramètres de ligne de commande `--export-map` et `--export-tileset`.

Quelques *Options d'Exportation* sont disponibles, qui sont appliquées aux cartes et jeux de tuiles avant qu'ils ne soient exportés (sans affecter la carte ou le jeu de tuiles lui-même).

14.1 Formats de Fichiers Génériques

Tiled supporte l'exportation vers plusieurs formats de fichiers génériques qui ne visent pas un environnement spécifique.

14.1.1 JSON

Le format JSON est le format de fichier supplémentaire le plus commun supporté par Tiled. Il peut être utilisé plutôt que le TMX depuis que Tiled peut ouvrir des cartes et jeux de tuiles en JSON et le format supporte toutes les fonctionnalités de Tiled. Le format JSON est plus facile à charger, surtout dans un navigateur et lors de l'utilisation de JavaScript en général.

14.1.2 Lua

Les cartes et jeux de tuiles peuvent être exportés en tant que code Lua. Cette option d'exportation supporte la majorité des fonctionnalités de Tiled et est utile lors de l'utilisation d'un environnement basé sur Lua tel que **LÖVE** (avec [Simple Implémentation avec Tiled](#)), **Solar2D** (avec `ponytiled` ou [Dusk Engine](#)) ou **Defold**.

Pour l'instant, les types de propriétés personnalisés ne sont pas implémentés (cependant le type affecte la façon dont la valeur de la propriété est exportée), ainsi que les informations liées aux [modèles d'objet](#).

The tiles are referenced using [Global Tile IDs](#), as done in the [TMX](#) and [JSON](#) formats.

14.1.3 CSV

L'exportation en CSV ne supporte que les [calques de tuiles](#). Les cartes contenant plusieurs calques de tuiles seront exportées en tant que plusieurs fichiers, nommés `base_<nom-du-calque>.csv`.

Chaque tuile est écrite en tant que son ID, sauf si la tuile a une propriété personnalisée nommée `nom`, où dans ce cas cette valeur sera utilisée pour écrire la tuile. L'utilisation de plusieurs jeux de tuiles va mener à des IDs ambigus, sauf si la propriété personnalisée `nom` est utilisée. Les tuiles vides ont une valeur de -1.

Quand les tuiles sont inversées horizontalement, verticalement ou diagonalement, ces états sont exportés en utilisant des drapeaux de bits dans l'ID, de la même façon que le [Format de Carte TMX](#).

14.2 Defold

Tiled peut exporter vers **Defold** en utilisant un des deux greffons fournis. Ils sont tous deux désactivés par défaut.

14.2.1 defold

Ce greffon exporte une carte en tant que [Carte de Tuiles Defold](#) (*.tilemap). Il supporte seulement des calques de tuiles et seulement un seul jeu de tuiles peut être utilisé.

Lors de l'exportation, la propriété `tile_set` de la carte de tuiles est laissée vide, donc elle doit être remplie dans Defold après chaque export.

14.2.2 defoldcollection

Ce greffon exporte une carte en tant que [Collection Defold](#) (*.collection), et crée aussi plusieurs fichiers .tilemap.

Il supporte :

- Les calques de groupes (**seuls les calques de groupes à la racine sont supportés, pas ceux qui sont imbriqués !**)
- Plusieurs Jeux de Tuiles par Carte de Tuiles

Lors de l'exportation :

- La propriété `Path` de chaque jeu de tuiles peut avoir à être modifié manuellement dans Defold après chaque exportation. Cependant, Tiled va essayer de trouver le fichier `.tilesource` correspondant avec le nom du jeu de tuiles dans Tiled dans le répertoire `/tilesources/` de votre projet. Si un fichier est trouvé, un ajustement manuel ne sera pas nécessaire.
- Si vous créez des propriétés personnalisées pour votre carte nommées `x-offset` et `y-offset`, ces valeurs seront utilisées en tant que coordonnées pour votre `GameObject` de haut niveau dans la collection. C'est utile si vous travaillez avec des [Mondes](#).

Tous les calques d'une carte de tuiles auront une propriété d'index Z assignée avec des valeurs fluctuant entre 0 et 0.1. Le greffon supporte l'utilisation de 9999 calques de groupes et 9999 calques de tuiles par calque de groupes.

Quand toute information supplémentaire est nécessaire pour la carte, elle peut être exportée dans un *format Lua* et chargée en tant que script Defold.

14.3 GameMaker : Studio 1.4

GameMaker : Studio 1.4 un format personnalisé basé sur le XML pour stocker ses salles, et Tiled est distribué avec un greffon qui permet d'exporter des cartes dans ce format. Pour le moment, seules les cartes orthogonales seront exportées correctement.

Les calques de tuiles et les objets tuiles (quand aucun type n'est donné) seront exportés en tant qu'éléments « tile ». Ceux-ci supportent l'inversion horizontale et verticale, mais pas de rotation. Pour les objets tuiles, l agrandissement est aussi supporté.

Avertissement : Les jeux de tuiles doivent être nommés de façon à être correspondantes aux arrières-plans dans le projet GameMaker. Sinon, GameMaker va afficher une erreur pour chaque tuile lors du chargement du fichier `room.gmx` exporté.

14.3.1 Instances d'Objet

Les instances d'objet de GameMaker sont créées en mettant le nom « object » dans le champ « Type » de l'objet dans Tiled. La rotation est supportée ici, et pour les objets tuiles les inversions et l agrandissement sont aussi supportés (cependant la combinaison de l'inversion et de la rotation n'a pas l'air de marcher dans GameMaker).

Les propriétés personnalisées suivantes peuvent être ajoutées aux objets pour affecter leur instance exportée :

- La chaîne de caractères (string) `code` (code de création d'instance, « » par défaut)
- Le flottant (float) `scaleX` (déduit de la tuile ou 1.0 par défaut)
- Le flottant (float) `scaleY` (déduit de la tuile ou 1.0 par défaut)
- L'entier (int) `originX` (0 par défaut)
- L'entier (int) `originY` (0 par défaut)

Les propriétés `scaleX` et `scaleY` peuvent être utilisées pour remplacer l agrandissement de l'instance. Cependant, si l agrandissement est important, alors il est généralement plus facile d'utiliser un objet tuile, qui dans ce cas sera automatiquement dérivé de la taille de la tuile et de la taille de l'objet.

Les propriétés `originX` et `originY` peuvent être utilisées pour donner l'origine de l'objet défini dans GameMaker à Tiled en tant que décalage depuis en haut à gauche. Cette origine est prise en compte lors de la détermination de la position de l'instance exportée.

Indication : Bien sûr, donner le type et/ou les propriétés au dessus manuellement pour chaque instance va vite devenir ennuyeux. Depuis Tiled 1.0.2, vous pouvez utiliser des objets tuile avec le type donné sur la tuile à la place, et dans Tiled 1.1 vous pouvez aussi utiliser des *modèles d'objet*.

14.3.2 Vues

Les vues peuvent être définies en utilisant des *objets rectangles* dont le Type a été changé en *view*. La position et la taille va être convertie en pixels. La visibilité de la vue au début de la salle dépend de la visibilité de l'objet. L'utilisation des vues est automatiquement activée quand toute vue est définie.

Les propriétés personnalisées suivantes peuvent être utilisées pour définir les propriétés variées de la vue :

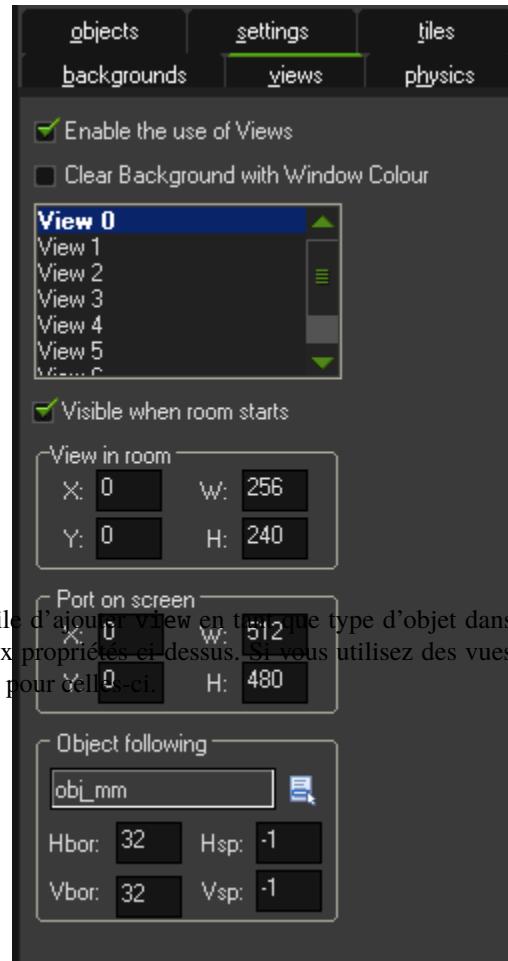
Position sur l'écran

- L'entier (int) **xport** (0 par défaut)
- L'entier (int) **yport** (0 par défaut)
- L'entier (int) **wport** (1024 par défaut)
- L'entier (int) **hport** (768 par défaut)

Suivi d'objet

- La chaîne de caractères (string) **objName**
- L'entier (int) **hborder** (32 par défaut)
- L'entier (int) **vborder** (32 par défaut)
- L'entier (int) **hspeed** (-1 par défaut)
- L'entier (int) **vspeed** (-1 par défaut)

Indication : Lorsque vous définissez des vues dans Tiled, il est utile d'ajouter *view* en tant que type d'objet dans l'*Éditeur de Types d'Objet*, ce qui permet d'avoir un accès facile aux propriétés ci-dessous. Si vous utilisez des vues avec des options similaires, vous pouvez mettre en place des *modèles* pour celle-ci.



14.3.3 Propriétés de Carte

Général

- L'entier (int) **speed** (30 par défaut)
- Le booléen (bool) **persistent** (false par défaut)
- Le booléen (bool) **clearDisplayBuffer** (true par défaut)
- Le booléen (bool) **clearViewBackground** (false par défaut)
- La chaîne de caractères (string) **code** (code de création de carte, « » par défaut)

Propriétés Physiques

- Le booléen (bool) **PhysicsWorld** (false par défaut)
- L'entier (int) **PhysicsWorldTop** (0 par défaut)
- L'entier (int) **PhysicsWorldLeft** (0 par défaut)
- L'entier (int) **PhysicsWorldRight** (longueur de la carte en pixels par défaut)
- L'entier (int) **PhysicsWorldBottom** (hauteur de la carte en pixels par défaut)
- Le flottant (float) **PhysicsWorldGravityX** (0.0 par défaut)
- Le flottant (float) **PhysicsWorldGravityY** (10.0 par défaut)
- Le flottant (float) **PhysicsWorldPixToMeters** (0.1 par défaut)

14.3.4 Propriétés des Calques

Les calques de tuiles et les calques d'objets peuvent tous deux produire des éléments « tile » dans le fichier de salle exporté. Leur profondeur est mise en place automatiquement, avec les tuiles du calque le plus en bas ayant une valeur de 10000000 (la valeur de GameMaker par défaut) et elle augmente à partir de cette valeur. Si vous voulez mettre en place une valeur de profondeur par défaut, vous pouvez changer la propriété suivante sur le calque :

- L'entier (int) `depth` (10000000 + N par défaut)

14.4 GameMaker Studio 2.3

GameMaker Studio 2.3 utilise un format basé sur le JSON pour stocker ses salles, et Tiled est distribué avec un greffon qui exporte les cartes dans ce format.

Ce greffon va faire de son mieux pour exporter la carte aussi fidèlement que possible, correspondant les fonctionnalités diverses de Tiled à celles de GameMaker. Les *calques de tuiles* sont exportés en tant calques de tuiles si possible, mais peuvent devenir des calques de ressources si nécessaire. Les *objets* peuvent être exportées en tant qu'instances, mais aussi en tant que graphismes de tuile, graphismes d'image ou vues. Les *calques d'images* sont exportés en tant que calques d'arrière-plan.

Avertissement : Since GameMaker's « Add Existing » action doesn't work at this point in time (2.3.1) the easiest way to export a Tiled map to your GameMaker Project is to overwrite an already existing `room.yy` file.

Starting with Tiled 1.8, it is no longer necessary to deactivate the « Use safe writing of files » option, since the GameMaker export now ignores it to avoid reload issues in GameMaker.

14.4.1 Référence aux Ressources Existantes

Comme Tiled ne fait qu'exporter une carte en tant que salle GameMaker pour le moment, toutes images, jeux de tuiles ou objets utilisés par la carte sont supposés être déjà existants dans le projet GameMaker.

Pour les images, le nom de l'image doit être dérivé du nom du fichier de l'image enlevant son extension de fichier. Si nécessaire, le nom de l'image peut être explicitement spécifié en une propriété personnalisée `sprite` (supportée pour les jeux de tuiles, les tuiles de jeux de collection d'images et les calques d'images).

Pour les jeux de tuiles, le nom du jeu de tuiles entré dans Tiled doit correspondre au nom du jeu de tuiles ressource dans GameMaker.

Pour les instances d'objet, le nom de l'objet doit être spécifié dans le champ `Type`.

14.4.2 Exporter une Carte Tiled

Une carte Tiled contient des calques de tuiles, des calques d'objets, des calques d'images et des groupes de calques. Tous ces types de calques sont supportés.

Calques de Tuiles

Si possible, une carte de tuiles va être exportée en tant que calque de tuiles.

Quand plusieurs jeux de tuiles sont utilisés dans le même calque, le calque est exporté en tant qu'un groupe avec un calque de tuiles enfant pour chaque jeu de tuiles, comme GameMaker ne supporte qu'un seul jeu de tuiles par calque de tuiles.

Quand la taille des tuiles d'un jeu de tuiles ne correspond pas à la taille de la grille de la carte, ou quand l'orientation de la carte n'est pas orthogonale (isométrique ou hexagonale, par exemple), les tuiles seront exportées en tant que calque de ressources à la place. Ce type de calque est plus flexible, cependant il ne supporte pas la rotation des graphismes des tuiles.

Quand le calque inclut des tuiles venant d'un jeu de tuiles de collection d'images, celles-ci seront exportées dans un calque de ressources en tant que graphismes d'image.

Calques d'Objets

Les calques d'objets dans Tiled sont très flexibles puisque les objets peuvent prendre un grand nombre de formes différentes. Pour cette raison, l'exportation examine chaque objet pour voir comment il doit être exporté dans la salle GameMaker.

Quand un objet a un *Type*, il est exporté en tant qu'instance sur un calque d'instance, où le type réfère au nom de l'objet à instancier. Cependant, si le type est « view », l'objet est interprété en tant que *vue*.

Quand un objet n'a aucun Type, mais que c'est un objet tuile, alors il est exporté soit en tant que graphisme de tuile ou graphisme d'image, dépendant de si la tuile appartient à une image de jeu de tuiles ou à une collection d'images.

Les propriétés personnalisées suivantes peuvent être ajoutées aux objets pour affecter l'instance exportée ou l'image ressource :

- La couleur (color) `colour` (basé sur la couleur de teinte du calque par défaut)
- Le flottant (float) `scaleX` (déduit de la tuile ou 1.0 par défaut)
- Le flottant (float) `scaleY` (déduit de la tuile ou 1.0 par défaut)
- Le booléen (bool) `inheritItemSettings` (false par défaut)
- L'entier (int) `originX` (0 par défaut)
- L'entier (int) `originY` (0 par défaut)
- Le booléen (bool) `ignore` (si l'objet est caché par défaut)

Les propriétés `scaleX` et `scaleY` peuvent être utilisées pour remplacer l'agrandissement de l'instance. Cependant, si l'agrandissement est important, alors il est généralement plus facile d'utiliser un objet tuile, qui dans ce cas sera automatiquement dérivé de la taille de la tuile et de la taille de l'objet.

Les propriétés `originX` et `originY` peuvent être utilisées pour donner l'origine de l'image définie dans GameMaker à Tiled en tant que décalage depuis en haut à gauche. Cette origine est prise en compte lors de la détermination de la position de l'instance exportée.

Indication : Bien sûr, donner le type et/ou les propriétés au dessus manuellement pour chaque instance va vite devenir ennuyeux. À la place, vous pouvez utiliser des objets tuile avec le type donné sur la tuile ou utiliser des *modèles d'objet*.

Instances d'Objet

Les propriétés personnalisées additionnelles suivantes peuvent être ajoutées à des objets qui sont exportés en tant qu'instances d'objet :

- Le booléen (bool) `hasCreationCode` (false par défaut)
- L'entier (int) `imageIndex` (0 par défaut)
- Le flottant (float) `imageSpeed` (1.0 par défaut)
- L'entier (int) `creationOrder` (0 par défaut)

La propriété `hasCreationCode` peut être mise à true. Cela réfère au fichier « `InstanceCreationCode_[nom_d'instance].gml` » dans le dossier de la salle que vous pouvez créer dans GameMaker lui-même ou avec un éditeur de texte externe.

Par défaut l'ordre de la création de l'instance dépend de la position des objets dans le calque et de la hiérarchie des objets dans Tiled. Cela peut être changé en utilisant la propriété personnalisée `creationOrder`. Les objets ayant des valeurs plus petites seront créés avant les objets ayant des valeurs plus grandes (donc les objets avec des valeurs négatives seront créés avant les objets sans propriété `creationOrder`).

Les propriétés personnalisées additionnelles qui ne sont pas documentées ici peuvent être utilisées pour remplacer les définitions des variables qui ont été mises en place dans GameMaker pour l'objet.

Note : Pour le moment seules les définitions de variables de l'objet lui-même peuvent être remplacées. Le remplacement de définitions de variables d'objets parents n'est pas supporté. Vous pouvez utiliser le code de création pour remplacer les variables d'un objet parent en tant que palliatif.

Graphismes de Tuile

Pour les objets exportées en tant que graphismes de tuile (tuiles GMS 1.4), il doit être noté que la rotation n'est pas supportée pour les calques de ressources.

Quand une rotation de 90 degrés avec un alignement de grille suffit, ces tuiles doivent être placées dans des calques de tuiles à la place. Quand un placement libre avec une rotation est requis, alors un jeu de tuiles de collection d'images doit être utilisé, pour que les objets exportés soient exportés en tant que graphismes d'image à la place.

Graphismes d'Image

Les propriétés personnalisées additionnelles suivantes peuvent être ajoutées aux objets qui sont exportés en tant que graphismes d'image :

- Le flottant (float) `headPosition` (0.0 par défaut)
- Le flottant (float) `animationSpeed` (1.0 par défaut)

Calques d'Images

Les calques d'images sont exportés en tant que calques d'arrière-plan.

Le nom de fichier de l'image source est présumé être le même que le nom de l'image ressource correspondante. Alternativement, la propriété personnalisée `sprite` peut être utilisée pour explicitement donner le nom de l'image ressource.

Même s'il n'y a pas de support visuel dans Tiled, il est possible de créer un calque d'images sans image mais seulement avec une couleur de teinte. De tels calques seront exportés en tant que calque d'arrière-plan avec juste la couleur mise en place.

Les propriétés personnalisées suivantes peuvent être ajoutées aux calques d'images pour affecter les calques d'arrière-plan exportés :

- La chaîne de caractères (string) `sprite` (basé sur le nom de fichier de l'image par défaut)
- Le booléen (bool) `htiled` (false par défaut)
- Le booléen (bool) `vtiled` (false par défaut)
- Le booléen (bool) `stretch` (false par défaut)
- Le flottant (float) `hspeed` (0.0 par défaut)
- Le flottant (float) `vspeed` (0.0 par défaut)
- Le flottant (float) `animationFPS` (15.0 par défaut)
- L'entier (int) `animationSpeedtype` (0 par défaut)

Même si les propriétés personnalisées telles que `htiled` et `vtiled` n'ont pas d'effet visuel dans Tiled, vous verrez l'effet dans la salle exportée dans GameMaker.

14.4.3 Cas Spéciaux et Propriétés Personnalisées

Salles

Si une Couleur d'Arrière-Plan est donnée dans les propriétés de la carte Tiled, un calque d'arrière-plan supplémentaire avec cette couleur sera exportée en tant que calque le plus en bas.

Les propriétés personnalisées suivantes peuvent être données dans *Carte -> Propriétés de Carte*.

Général

- La chaîne de caractères (string) `parent` (« Rooms » par défaut)
- Le booléen `inheritLayers` (false par défaut)
- La chaîne de caractères (string) `tags` (« » par défaut)

La propriété `parent` est utilisée pour définir le dossier parent dans le navigateur de ressources de GameMaker.

La propriété `tags` est utilisée pour assigner des étiquettes à la salle. Plusieurs étiquettes doivent être séparées par une virgule.

Options de Salle

- Le booléen `inheritRoomSettings` (false par défaut)
- Le booléen (bool) `persistent` (false par défaut)
- Le booléen (bool) `clearDisplayBuffer` (true par défaut)
- Le booléen `inheritCode` (false par défaut)
- La chaîne de caractères (string) `creationCodeFile` (« » par défaut)

La propriété `creationCodeFile` est utilisée pour définir le chemin vers un fichier de code de création existant, par exemple : « \${project_dir}/rooms/room_name/RoomCreationCode.gml ».

Fenêtres d'Affichage et Caméras

Général

- Le booléen `inheritViewSettings` (false par défaut)
- Le booléen `enableViews` (true si tout objet « view » a été trouvé par défaut)
- Le booléen (bool) `clearViewBackground` (false par défaut)

Fenêtre d'Affichage 0 - Fenêtre d'Affichage 7

Vous pouvez configurer jusqu'à 8 fenêtres d'affichage en utilisant des objets vue (voir [Vues](#)).

Propriétés Physiques

- Le booléen `inheritPhysicsSettings` (false par défaut)
- Le booléen (bool) `PhysicsWorld` (false par défaut)
- Le flottant (float) `PhysicsWorldGravityX` (0.0 par défaut)
- Le flottant (float) `PhysicsWorldGravityY` (10.0 par défaut)
- Le flottant (float) `PhysicsWorldPixToMeters` (0.1 par défaut)

Références d'Images

Comme [mentionné ci-dessus](#), les références vers des images sont généralement dérivées du nom de la ressource d'image depuis le nom du fichier d'image. Les propriétés suivantes peuvent être données sur des jeux de tuiles, des tuiles d'un jeu de tuiles de collection d'images et des calques d'images pour explicitement spécifier le nom de l'image :

- La chaîne de caractères (string) `sprite` (basé sur le nom de fichier de l'image par défaut)

Chemins

Avertissement : Les chemins ne sont pas encore supportés, mais il est prévu d'exporter les objets polyligne et polygones en tant que chemins sur des calques de chemins dans une future mise à jour.

Vues

Les vues peuvent être définies en tant qu'[objets rectangles](#) où le *Type* a été changé en « view ». La position et la taille seront converties en pixels. La visibilité de la vue quand la salle commence dépend de la visibilité de l'objet. L'utilisation de vues est automatiquement activée si toute vue est définie.

Les propriétés personnalisées suivantes peuvent être utilisées pour définir les propriétés variées de la vue :

Général

- Le booléen (bool) ``inherit`` (false par défaut)

Propriétés de la Caméra

Les Propriétés de la Caméra sont automatiquement dérivées de la position et de la taille de l'objet vue.

Propriétés de la Fenêtre d'Affichage

- L'entier (int) `xport` (0 par défaut)
- L'entier (int) `yport` (0 par défaut)
- L'entier (int) `wport` (1366 par défaut)
- L'entier (int) `hport` (768 par défaut)

Suivi d'objet

- La chaîne de caractères `objectId`
- L'entier (int) `hborder` (32 par défaut)
- L'entier (int) `vborder` (32 par défaut)
- L'entier (int) `hspeed` (-1 par défaut)
- L'entier (int) `vspeed` (-1 par défaut)

Indication : Lorsque vous définissez des vues dans Tiled, il est utile d'ajouter `view` en tant que type d'objet dans [l'Éditeur de Types d'Objet](#), ce qui permet d'avoir un accès facile aux propriétés ci-dessus. Si vous utilisez des vues avec des options similaires, vous pouvez mettre en place des [modèles](#) pour celles-ci.

Calques

Tous les types de calques supportent les propriétés personnalisées suivantes :

- L'entier (int) **depth** (assigné automatiquement par défaut, comme dans GameMaker)
- Le booléen (bool) **visible** (dérivé du calque par défaut)
- Le booléen (bool) **hierarchyFrozen** (l'état de verrouillage du calque par défaut)
- Le booléen (bool) **noExport** (false par défaut)

La propriété **depth** peut être utilisée pour assigner une valeur de profondeur spécifique à un calque.

La propriété **visible** peut être utilisée pour remplacer l'état « Visible » du calque si besoin.

La propriété **hierarchyFrozen** peut être utilisé pour remplacer l'état « Verrouillé » du calque si besoin.

La propriété **noExport** peut être utilisée pour annuler l'exportation d'un calque en entier, incluant tous ses calques enfants. C'est utile si vous utilisez un calque pour des commentaires (comme l'addition d'une image d'arrière-plan ou d'objets texte) que vous ne voulez pas exporter dans GameMaker. Veuillez noter que toute vue définie dans ce calque sera aussi ignorée.

14.5 tBIN

Le format de carte tBIN est un format binaire utilisé par l'Éditeur de Carte de Tuiles tIDE. tIDE est utilisé par *Stardew Valley*, un jeu à succès qui a amené la création de nombreux modules créés par la communauté.

Tiled est distribué avec un greffon qui active l'édition directe de cartes de Stardew Valley (et de toute autre carte utilisant le format tBIN). Ce greffon doit être activé dans *Édition > Préférences > Greffons*. Il n'est pas activé par défaut car il ne sauvegardera pas tout (surtout il ne supporte pas les calques d'objets en général, ni les jeux de tuiles externes), donc vous devez savoir ce que vous êtes en train de faire.

Note : Le format tBIN supporte l'utilisation de propriétés personnalisées sur les tuiles d'un calque de tuiles. Comme Tiled ne le supporte pas directement, des objets « TileData » sont créés qui correspondent à la position de la tuile, sur lesquels ces propriétés sont stockées.

14.6 Autres Formats

D'autres greffons fournis avec Tiled supportent des jeux ou outils variés :

droidcraft Ajoute un support pour éditer des cartes DroidCraft (*.dat)

flare Ajoute un support pour éditer des cartes de Flare Engine (*.txt)

replicaisland Ajoute un support pour éditer des cartes de Replica Island (*.bin)

rpmap Ajoute un support pour exporter des cartes Tiled en tant que RpMap (*.rpmap), le format utilisé par [Map-Tool](#).

Le support est limité aux cartes utilisant des jeux de tuiles « Collection d'Images » pour le moment vu que MapTool ne supporte pas les images de jeu de tuiles.

tengine Ajoute un support pour exporter en tant que carte *T-Engine* (*.lua)

Ces greffons sont désactivés par défaut. Ils peuvent être activés dans *Édition > Préférences > Greffons*.

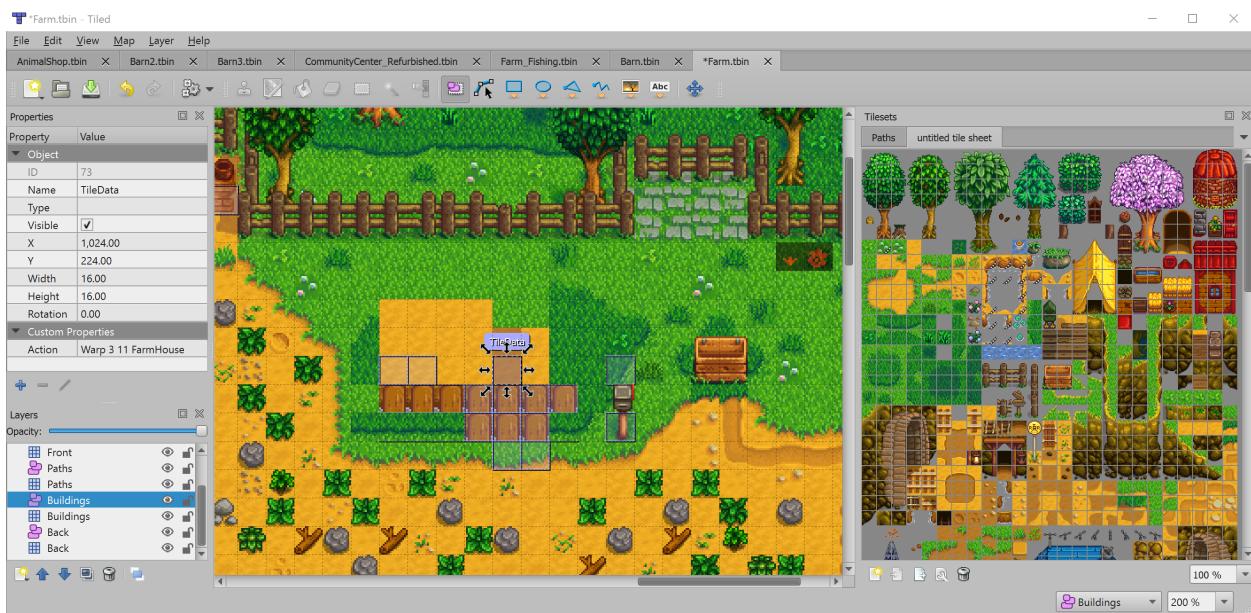


Fig. 1 – Une des cartes de ferme de Stardew Valley ouverte dans Tiled.

14.7 Formats d'Exportation Personnalisés

Tiled offre plusieurs options pour être étendu avec un support pour des formats de fichiers supplémentaires.

14.7.1 Utiliser JavaScript

Tiled est *extensible en utilisant JavaScript* et il est possible d'ajouter de nouveaux formats en utilisant `tiled.registerMapFormat` ou `tiled.registerTilesetFormat`.

14.7.2 Utiliser Python

Il est aussi possible d'écrire des *scripts Python* pour ajouter un support pour l'importation et l'exportation de formats de cartes personnalisés.

14.7.3 Utiliser C++

Pour l'instant, toutes les options d'exportation distribuées avec Tiled sont écrites en tant que greffons de Tiled en C++. L'API pour de tels greffons n'est pas documentée (mis à part des commentaires dans le style de Doxygen dans le code source de `libtiled`), mais il y a plus d'une douzaine d'exemples que vous pouvez étudier.

Note : Pour des raisons de compatibilité binaire, un greffon en C++ a besoin d'être compilé pour la même plateforme, par le même compilateur et avec les mêmes versions de Qt et de Tiled que le plugin est censé supporter. Généralement, le moyen le plus facile d'y arriver est de compiler le greffon en même temps que Tiled, ce qui est fait avec tous les greffons actuels. Si vous écrivez un greffon en C++ qui peut être utile pour d'autres utilisateurs, il est recommandé d'ouvrir une requête de fusion pour qu'il soit distribué avec Tiled.

14.8 Exporter en Tant qu'Image

Les cartes peuvent être exportées en tant qu'image. Tiled supporte la plupart des formats d'image communs. Choisissez *fichier -> Exporter en Tant qu'Image...* pour ouvrir la boîte de dialogue concernée.

Comme l'exportation de carte résulte parfois en une image énorme, l'utilisation de l'option *Utiliser le niveau de zoom actuel* est donnée pour permettre l'exportation de la carte dans la taille à laquelle elle est couramment affichée.

Si vous voulez convertir une carte en tant qu'image de façon répétée, lancer l'exportation manuellement n'est pas très pratique. Pour ce cas-ci, un outil appelé `tmxrasterizer` est distribué avec Tiled, qui permet de générer un rendu de n'importe quel format de carte supporté en tant qu'image, contrairement à son nom. Il est aussi capable de générer un rendu pour des *mondes entiers* en tant qu'image. Sur Linux, cet outil peut être utilisé pour générer des miniatures de prévisualisation de cartes pour le gestionnaire de fichier.

Note : Lors d'une exportation via ligne de commande sur Linux, Tiled aura quand même besoin d'un serveur X pour fonctionner. Pour automatiser des exportations dans un environnement sans interface graphique, vous pouvez utiliser un serveur X sans interface graphique tel que [Xvfb](#). Dans ce cas-ci, vous voulez lancer Tiled à travers une ligne de commande comme ceci :

```
xvfb-run tiled --export-map ...
```

CHAPITRE 15

Raccourcis Clavier

Note : La majorité des raccourcis clavier ci-dessous peuvent être changés dans les *Préférences*.

Sur macOS, remplacez Ctrl ` par la touche ``Command.

15.1 Général

- Ctrl + N - Créer une nouvelle carte
- Ctrl + O - Ouvrir n'importe quel fichier ou projet
- Ctrl + P - Ouvrir un fichier dans le projet courant
- Ctrl + Maj + T - Rouvrir un fichier récemment fermé
- Ctrl + S - Sauvegarder le document actuel
- Ctrl + Alt + S - Sauvegarder le document actuel vers un autre fichier
- Ctrl + Maj + S - Sauvegarder tous les documents
- Ctrl + E - Exporter le document actuel
- Ctrl + Maj + E - Exporter le document actuel vers un autre fichier
- Ctrl + R - Recharger le document actuel
- Ctrl + W - Fermer le document actuel
- Ctrl + Maj + W - Fermer tous les documents
- Ctrl + Q - Arrêter Tiled
- Ctrl + Roulette - Zoomer en avant/arrière sur le jeu de tuiles et la carte
- Ctrl + Plus/Moins - Zoomer en avant/arrière sur la carte
- Ctrl + Ø - Réinitialiser le zoom sur la carte
- Ctrl + / - Ajuster le zoom pour que la carte rentre dans la vue
- Ctrl + Mouvement d'un Objet - Activer temporairement ““Aligner sur la Grille””
- Ctrl + Changement de Taille d'un Objet - Conserver les proportions de l'objet
- Alt + Changement de Taille d'un Objet - Activer temporairement ““Aligner sur la Grille””
- Click Milieu ou Barre Espace - Maintenir pour afficher la vue cartographique
- Ctrl + X - Couper (tuiles, objets ou propriétés)
- Ctrl + C - Copier (tuiles, objets ou propriétés)

- Ctrl + V - Coller (tuiles, objets ou propriétés)
- Del - Supprimer (tuiles, objets, propriétés ou calques)
- Ctrl + G - Activer / Désactiver l'affichage de la grille de tuile
- H - Basculer la brillance du calque actuel
- Ctrl + M - Invoyer l'*Automapping*
- Alt + C - Copier la position actuelle du curseur de la souris dans le presse-papiers (en tant que coordonnées de tuiles)
- Ctrl + D - Dupliquer les objets sélectionnés
- Ctrl + J - Créer un nouveau calque et copier les objets et tuiles couramment sélectionnés dans celui-ci
- Ctrl + Maj + J - Créer un nouveau calque et déplacer les objets et tuiles couramment sélectionnés dans celui-ci
- Ctrl + Maj + D - Dupliquer les calques sélectionnés
- F2 - Renommer (si applicable dans le contexte)
- Tab - Cacher les fenêtres subsidiaires et les barres d'outils
- Ctrl + PgHaut - Sélectionner le calque précédent (au-dessus du calque actuel)
- Ctrl + PgBas - Sélectionner le prochain calque (en-dessous du calque actuel)
- Ctrl + Maj + Haut - Monter les calques sélectionnés
- Ctrl + Maj + Bas - Descendre les calques sélectionnés
- Ctrl + H - Afficher/Cacher les calques sélectionnés
- Ctrl + L - Verrouiller/Déverrouiller les calques sélectionnés
- Ctrl + Maj + H - Montrer/Cacher tous les autres calques (seul le calque actif reste visible/tous les calques deviennent visible)
- Ctrl + Maj + L - Verrouiller/Déverrouiller tous les autres calques
- Ctrl + Tab / Alt + Gauche - Passer au document à gauche
- Ctrl + Maj + Tab / Alt + Droite - Passer au document à droite
-] - Sélectionner le jeu de tuiles suivant
- [- Sélectionner le jeu de tuiles précédent
- Ctrl + T - Forcer le recharge de tous les jeux de tuiles utilisés par la carte actuelle (principalement utilisé quand le recharge automatique n'est pas activé)
- Ctrl + Maj + A - Vider toute sélection d'objet ou de tuile

15.2 Quand un calque de tuiles est sélectionné

- Clic Droit sur une Tuile - Copier la tuile sous la souris (glisser pour copier des zones plus grandes).
- Ctrl + Clic Droit sur une Tuile - Sélectionner le calque contenant la tuile la plus en haut sous la souris.
- D - Activer/Désactiver le Mode Aléatoire
- B - Activer la *Brosse Tampon*
 - Maj + Click - Mode ligne, permet de placer des tuiles sur une ligne entre deux positions cliquées
 - Ctrl + Maj + Click - Mode cercle, permet de placer des tuiles autour du centre cliqué
- T - Activer la *Brosse de Terrain*
- F - Activer l'*Outil de Seau de Remplissage*
- P - Activer l'*Outil de Remplissage de Forme*
- E - Activer la *Gomme*
- R - Activer la Sélection Rectangulaire
- W - Activer la Baguette Magique
- S - Activer la Sélection de Même Tuile
- Ctrl + 1-9 - Stocker la sélection de tuile courante. Quand aucun outil de dessin de tuile n'est sélectionné, il essaie de capturer la sélection de tuile courante (similaire à Ctrl + C).
- 1-9 - Rappeler un tampon de tuiles stocké précédemment (similaire à Ctrl + V)
- Ctrl + A - Sélectionner le calque en entier

Changer le tampon actuel :

- X - Inverser le tampon actif horizontalement

- Y - Inverser le tampon actif verticalement
- Z - Pivoter le tampon actif dans le sens des aiguilles d'une montre
- Maj + Z - Pivoter le tampon actif dans le sens inverse des aiguilles d'une montre

15.3 Quand un calque d'objets est sélectionné

- S - Activer l'outil pour *Sélectionner des Objets*
 - PgHaut - Déplacer les objets sélectionnés vers le haut (avec l'ordre d'affichage d'objet manuel)
 - PgBas - Déplacer les objets sélectionnés vers le bas (avec l'ordre d'affichage d'objet manuel)
 - Menu - Déplacer les objets sélectionnés tout en haut (avec l'ordre d'affichage d'objet manuel)
 - Fin - Déplacer les objets sélectionnés tout en bas (avec l'ordre d'affichage d'objet manuel)
- O - Activer l'*Éditer des Polygones*
- R - Activer l'outil pour *Insérer un Rectangle*
- I - Activer l'outil pour *Insérer un Point*
- C - Activer l'outil pour *Insérer une Ellipse*
- P - Activer l'outil pour *Insérer un Polygone*
 - Entrée - Finir la création de l'objet
 - Échap - Annuler la création de l'objet
- T - Activer l'outil pour *Insérer une Tuile*
- V - Activer l'outil pour *Insérer un Modèle* (depuis Tiled 1.1)
- E - Activer l'outil pour *Insérer un Texte*
- Ctrl + A - Sélectionner tous les objets des calques sélectionnés

15.4 Dans la Boîte de Dialogue de Propriétés

- Retour Arrière - Supprimer une propriété

CHAPITRE 16

Préférences Utilisateur

Il n'y a que quelques options dans les Préférences qui sont accessibles à travers le menu *Édition > Préférences*. La plupart des autres options, tel que si la grille est dessinée, le type de suivi à faire ou les dernières options utilisées lors de la création d'une nouvelle carte sont simplement gardés de façon persistante.

Les préférences sont stockées dans un format et emplacement dépendant du système :

Windows	La clé de registre HKEY_CURRENT_USER\SOFTWARE\mapeditor.org\Tiled
macOS	~/Library/Preferences/org.mapeditor.Tiled.plist
Linux	~/.config/mapeditor.org/tiled.conf

16.1 Général

16.1.1 Sauvegarde et Chargement

Recharger les images de jeu de tuiles lorsqu'elles changent

C'est très utile si vous travaillez avec des tuiles ou si les tuiles peuvent changer à cause du résultat d'un système de contrôle de source.

Rétablissement la session précédente lors du démarrage

Lorsque ceci est désactivé, Tiled commencera toujours avec une session vide. Cela peut être utile quand vous changez de projet souvent.

Utiliser l'écriture sécurisée des fichiers

Cette option force les fichiers à être écrits dans un fichier temporaire, et si tout est bien allé, d'être interchangé avec le fichier cible. Cela évite une perte de données à cause d'erreurs lors de la sauvegarde ou d'espace disque insuffisant. Malheureusement, cela peut causer des problèmes lors de la sauvegarde de fichiers sur un dossier Dropbox ou un disque en ligne, dans ce cas il peut être utile de désactiver cette fonctionnalité.

Répéter la dernière exportation lors de la sauvegarde

Quand cette fonctionnalité est activée, à chaque fois que vous sauvegardez une carte ou un jeu de tuiles qui a été exporté précédemment, il sera exporté une nouvelle fois au même endroit et dans le même format.

16.1.2 Options d'Exportation

Les options d'exportation suivantes sont appliquées à chaque fois qu'une carte ou jeu de tuiles est exporté, sans affecter la carte ou le jeu de tuiles lui-même.

Intégrer les jeux de tuiles

Tous les jeux de tuiles sont intégrés dans la carte exportée. C'est utile pour si par exemple vous voulez exporter faire un export en JSON et le chargement d'un jeu de tuiles externe n'est pas désiré.

Détacher les modèles

Toutes les instances de modèles sont détachées. C'est utile si vous voulez utiliser la fonctionnalité de modèles

mais vous ne pouvez ou voulez pas charger les fichiers d'objets modèles externes.

Résoudre les types et propriétés d'objet

Stocke les types et propriétés d'objet effectives dans chaque objet. Les propriétés d'objet sont héritées d'une tuile (dans le cas d'un objet tuile) et des propriétés pas défaut de leur type.

Minimiser la sortie

Évite l'ajout de blancs non nécessaires dans le fichier de sortie. Cette option est supportée pour les formats XML (TMX et TSX), JSON et Lua.

Ces options sont aussi disponibles en tant qu'options lors de l'exportation en utilisant la ligne de commande.

16.2 Interface

16.2.1 Interface

Langue Par défaut la langue essaye d'utiliser celle du système, mais si elle choisit la mauvaise vous pouvez toujours la changer ici.

Couleur de la grille Car le noir n'est pas toujours la meilleure couleur pour la grille.

Division de la grille fine La grille de tuiles peut être divisée encore plus en utilisant cette option, ce qui affecte l'option « Suivre la Grille Fine » dans le menu *Vue > Suivi de Grille*.

Épaisseur de la ligne des objets

Les formes sont rendues avec une ligne de 2 pixels d'épaisseur par défaut, mais certaines personnes aiment avoir une ligne plus

fine ou même plus épaisse. Sur certains systèmes le redimensionnement basé sur le PPP va aussi affecter cette option.

Comportement de la sélection des objets

Par défaut l'outil *Sélectionner des Objets* sélectionne les objets de n'importe quel calque. Avec cette option, vous pouvez forcer une préférence de sélection d'objets dans les calques couramment sélectionnés, ou de seulement choisir des objets dans les calques sélectionnés.

Quand l'option « Surligner le Calque Courant » est activée, Tiled préfère automatiquement la sélection d'objets depuis les calques couramment sélectionnés.

Utilisation de l'accélération matérielle (OpenGL)

Ceci active une façon assez non-optimisée de rendre la carte en utilisant OpenGL. Ce n'est souvent pas une amélioration et cela peut mener à des plantages, mais dans certains scénarios cela peut rendre l'édition plus réactive.

La molette de la souris zoome par défaut

Cette option force la molette de la souris à zoomer sans avoir besoin de maintenir la touche Contrôle (ou Commande sur macOS). Cela peut être un moyen utile de naviguer sur la carte, mais cela peut aussi interférer avec le déplacement sur un pavé tactile.

Le clic milieu de la souris active le défilement automatique

Quand cette option est activée, le clic milieu de la souris ne glisse pas la carte directement mais contrôle la vitesse d'un mouvement continu à la place.

Utiliser un défilement doux

Cette option affecte le comportement lors du défilement avec les flèches directionnelles. Quand elle est désactivée, la vue défile en pas basé sur les événements d'appui de touche. Quand elle est activée (par défaut), le vue défile continuellement tant que la touche est maintenue.

16.2.2 Mises à Jour

Par défaut, Tiled vérifie les nouveautés et si de nouvelles versions existent et souligne toutes mises à jour dans la barre d'état. Vous pouvez désactiver cette fonctionnalité ici. Il est recommandé de garder au moins une de ces options activées.

Si vous désactivez l'affichage de nouvelles versions, vous pouvez toujours vérifier manuellement si une nouvelle version est disponible en ouvrant la boîte de dialogue *À Propos de Tiled*.

16.3 Clavier

Ici vous pouvez ajouter, enlever ou changer les raccourcis clavier de la majorité des actions.

Les raccourcis conflictuels sont en rouge. Ils ne marcheront que si vous résolvez le conflit.

Si vous personnalisez plusieurs raccourcis, il est recommandé d'utiliser la fonctionnalité d'exportation pour sauvegarder vos raccourcis quelque part,

pour que vous puissiez récupérer votre configuration ou la copier sur d'autres installations de Tiled.

16.4 Thème

Sur Windows et Linux, le style utilisé par défaut par Tiled est « Tiled Fusion ». C'est une version personnalisée du style « Fusion » qui est distribué avec Qt. Sur macOS, ce style peut aussi être utilisé, mais le style par défaut est « Native » car il a n'a vraiment pas l'air à sa place.

Le style « Tiled Fusion » permet la personnalisation de la couleur de base. Si vous choisissez une couleur de base foncée, le texte devient automatiquement blanc et quelques autres ajustements sont faits pour que le tout soit lisible. Vous pouvez aussi choisir une couleur de sélection personnalisée.

Le style « Native » essaye de s'accorder avec le système d'exploitation, et est disponible car il est dans quelques cas préférable au style personnalisé. La couleur de base et la couleur de sélection ne peuvent pas être changés en utilisant ce style car elles dépendent du système.

16.5 Greffons

Ici vous pouvez choisir quels greffons sont actifs, ainsi qu'ouvrir le dossier *d'extensions scriptées*.

Les greffons ajoutent un support pour des formats de fichiers de cartes et/ou de jeux de tuiles. Quelques greffons génériques sont activés par défaut, quand ceux qui sont un peu plus spécifiques doivent être activés manuellement.

Il n'y a pas besoin de redémarrer Tiled lors de l'activation ou de la désactivation de greffons. Quand le chargement d'un greffon échoue, essayez de passer votre souris sur son icône pour voir si l'infobulle affiche un message d'erreur utile.

Voir [Exporter des Formats](#) pour plus d'informations sur les formats de fichier supportés.

CHAPITRE 17

Scripts Python

Note : Depuis Tiled 1.3, Tiled peut être étendu en utilisant du *JavaScript*. L’API JavaScript fournit plus d’opportunités d’étendre les fonctionnalités de Tiled que l’ajout de formats de cartes personnalisés. Elle est complètement documentée et marche directement sur toutes les plateformes. Elle doit être préférée au greffon en Python si possible.

Tiled est distribué avec un greffon qui vous permet d’utiliser du Python 3 pour ajouter un support pour des formats de cartes personnalisés. C’est utile surtout car vous n’avez pas besoin de compiler Tiled vous-même et les scripts sont faciles à déployer sur toutes les plateformes.

Pour que les scripts soient chargés, ils doivent être placé dans `~/.tiled`. Tiled vérifie si ce répertoire change, donc il n’y a pas besoin de relancer Tiled après avoir ajouté ou changé des scripts (cependant le répertoire doit exister avant de lancer Tiled).

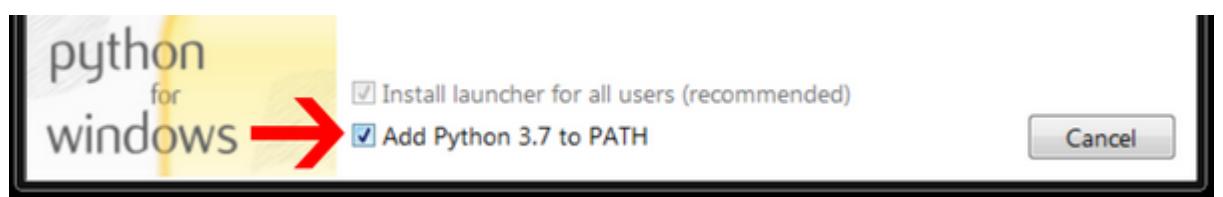
Il y a plusieurs scripts d’exemple disponibles dans le dépôt.

Note : Pour créer le dossier `~/.tiled` sur Windows, ouvrez l’invite de commande (`cmd.exe`), qui doit commencer dans votre dossier personnel par défaut, puis entrez `mkdir .tiled` pour créer le dossier.

Sur Linux, les dossiers commençant avec un point sont cachés par défaut. Dans la plupart des explorateurs de fichiers vous pouvez activer la visibilité des fichiers cachés en utilisant `Ctrl+H`.

Note : Depuis Tiled 1.2.4, les greffons en Python sont désactivés par défaut, car le chargement de ce greffon peut causer un plantage dépendant de la version de Python installée sur le système (#2091). Pour utiliser le greffon en Python, veuillez tout d’abord l’activer dans les Préférences.

Avertissement : Sur Windows, Python n’est pas installé par défaut. Pour que le greffon en Python de Tiled marche, il faut installer Python 3.7 (récupérez-le depuis <https://www.python.org/>). Vous aurez aussi besoin de cocher la case « Add Python 3.7 to PATH » (Ajouter Python 3.7 à PATH) dans le programme d’installation :



Sur Linux vous aurez aussi besoin d'installer le paquet approprié. Cependant, les versions de Linux courantes faites sur Ubuntu 18.04 utilisent Python 3.6, donc vous aurez besoin d'installer la nouvelle version d'une certaine manière.

Le greffon en Python n'est couramment pas activé pour les versions de macOS. Nous devons trouver comment utiliser Python 3, tandis que macOS ne distribue que Python 2.7 par défaut. Si vous utilisez ce greffons sur macOS vous aurez besoin d'utiliser Tiled 1.1 pour le moment.

17.1 Exemple de Greffon d'Exportation

Supposez que vous voulez une carte exportée dans le format suivant :

```
29,29,29,29,29,29,32,-1,34,29,29,29,29,29,29,  
29,29,29,29,29,32,-1,34,29,29,29,29,29,29,  
29,29,29,29,29,32,-1,34,29,29,29,29,29,29,  
29,29,29,29,29,32,-1,34,29,29,29,29,29,29,  
25,25,25,25,25,44,-1,34,29,29,29,29,29,29,  
-1,-1,-1,-1,-1,-1,34,29,29,29,29,29,29,  
41,41,41,41,41,41,42,29,29,24,25,25,25,  
29,29,29,29,29,29,29,29,29,29,32,-1,-1,-1,  
29,29,29,29,29,39,29,29,29,29,32,-1,35,41,  
29,29,29,29,29,29,29,29,29,29,32,-1,34,29,  
29,29,29,29,29,29,37,29,29,32,-1,34,29;
```

Vous pouvez réalisez ceci en sauvegardant le script `example.py` suivant dans le dossier de scripts :

```
from tiled import *

class Example(Plugin):
    @classmethod
    def nameFilter(cls):
        return "Example files (*.example)"

    @classmethod
    def shortName(cls):
        return "example"

    @classmethod
    def write(cls, tileMap, fileName):
        with open(fileName, 'w') as fileHandle:
            for i in range(tileMap.layerCount()):
                if isTileLayerAt(tileMap, i):
                    tileLayer = tileLayerAt(tileMap, i)
                    for y in range(tileLayer.height()):
```

(suite sur la page suivante)

(suite de la page précédente)

```

tiles = []
for x in range(tileLayer.width()):
    if tileLayer.cellAt(x, y).tile() != None:
        tiles.append(str(tileLayer.cellAt(x, y).tile().id()))
    else:
        tiles.append(str(-1))
line = ','.join(tiles)
if y == tileLayer.height() - 1:
    line += ';'
else:
    line += ','
print(line, file=fileHandle)

return True

```

Ensuite vous pourrez voir une entrée « Example files » dans la liste de types dans *Fichier > Exporter*, qui vous permet d'exporter la carte en utilisant le script ci-dessus.

Note : Cet exemple ne supporte pas l'utilisation des groupes de calques, et en fait l'API de création de script ne les supportent pas encore. Toute aide pour maintenir le greffon en Python serait vraiment appréciée. Voir [tickets ouverts liés au support Python](#).

17.2 Déboguer Votre Script

Toute erreur rencontrée lors de l'analyse ou du lancer du script sont affichées dans la Console, qui peut être activée dans *Vue > Vues et Barres d'Outils > Console*.

17.3 Référence de l'API

Ce serait bien d'avoir une référence complète de l'API documentée ici, mais pour le moment veuillez voir le [fichier source](#) pour les classes et méthodes disponibles.

CHAPITRE 18

Bibliothèques et Environnements

Il y a beaucoup de bibliothèques disponibles pour lire et/ou écrire des cartes Tiled (stockées au format *Format de Carte TMX* ou *json-map-format*) ainsi que de nombreux environnements de développement qui incluent un support pour les cartes Tiled. Cette liste est divisée en deux sections :

- *Support par Langage*
- *Support par Environnement*

La première liste est pour les développeurs qui ont prévu d'implémenter leur propre générateur de rendu. La seconde liste est pour les développeurs qui utilisent déjà (ou envisagent d'utiliser) un moteur de jeu particulier / des bibliothèques graphiques et qui voudraient plutôt éviter de devoir écrire leur propre générateur de rendu de carte de tuiles.

Note : Pour apporter des mises à jours à cette page, veuillez ouvrir une pull request (demande de tirage) ou un rapport de bug sur [GitHub](#), merci !

18.1 Support par Langage

Typiquement, ces bibliothèques incluent seulement un analyseur TMX, mais aucun support de rendu. Elles peuvent être utilisées universellement et ne devraient pas nécessiter de moteur de jeu particulier ou de bibliothèque graphique.

18.1.1 C

- [cute tiled](#) - Chargeur de cartes JSON avec des exemples (zlib/Domaine Public).
- [TMX](#) - Exemples de chargeurs de carte TMX avec Allegro5 et SDL2 (BSD).

18.1.2 C++

- [tmxpather](#) basé sur du C++/TinyXML (BSD)
- C++/Qt basé sur libtiled, utilisé par Tiled lui-même et inclue dans [src/libtiled](#) (BSD)
- [libtmx-parser](#) basé sur du C++11x/TinyXml2 par halsafar. (zlib/tinyxml2)
- [libtmx](#) basé sur du C++11/TinyXml2 par jube, uniquement pour lire (licence ISC). Voir la [documentation](#).
- [TMXParser](#) Chargeur de données de jeux de tuiles *.tmx général. Prévu pour être utilisé avec TSXParser pour un chargement extérieur de jeu de tuiles. (Pas de support interne de jeu de tuiles)
- [TSXParser](#) Chargeur de données de jeu de tuiles *.tsx général. Prévu pour être utilisé avec TMXParser.
- [TMXLoader](#) basé sur [RapidXml](#). Fonctionnalités limitées (visitez le [site web](#) pour plus de détails).
- [tmxlite](#) Analyseur de carte en C++14 avec support pour cartes compressées mais aucun lien extérieur n'est requis. Inclut des exemples pour des rendus SFML et SDL2. Actuellement il y a un support complet pour tmx à jour jusqu'à la 0.16. (Zlib/libpng)
- [tinytmx](#) A C++17 library to parse maps generated by Tiled Map Editor. Requires no external linking, all dependencies are included.
- [Tileson](#) - Un analyseur de JSON de Tiled pour du C++ moderne (C++17) par Robin Berg Pettersen (BSD)

18.1.3 C#/.NET

- [TiledCS](#) : Une bibliothèque dotnet pour charger des cartes et jeux de tuiles Tiled (TMX/TSX ou JSON).
- [MonoGame.Extended](#) possède un chargeur de carte Tiled et un générateur de rendu qui fonctionne avec MonoGame sur toutes les plateformes supportant les classes de bibliothèques portables.
- Les projets suivants n'ont plus l'air d'être maintenus, mais ils peuvent être quand même utiles : [TiledSharp](#), [NTiled](#), [tmx-mapper-pcl](#), [tiled-xna](#) et [TmxCSharp](#).

18.1.4 Clojure

- [tile-soup](#) : Analyse et valide un fichier TMX dans une carte. Décode automatiquement les données formatées en Base64 ou en CSV et constraint les nombres quand nécessaire. Marche sur la JVM et dans les navigateurs via ClojureScript.

18.1.5 D

- [tiledMap.d](#) un exemple simple d'un seul calque et d'un seul jeu de tuiles pour charger une carte et ses jeux de tuiles en [langage D](#). Il contient aussi un rendu logique et basique en utilisant [DSFML](#)
- [dtiled](#) peut charger des cartes Tiled formatées en JSON. Il fournit aussi des fonctions et algorithmes généraux liés aux jeux de tuiles.

18.1.6 Go

- [github.com/lafriks/go-tiled](#)
- [github.com/salviati/go-tmxtmx](#)

18.1.7 Haskell

- `htiled` (TMX) par Christian Rødli Amble.
- `aeson-tiled` (JSON) par Schell Scivally.

18.1.8 Java

- Une bibliothèque pour charger des fichiers TMX est inclue avec Tiled dans `util/java/libtiled-java`.
- `TiledReader` est un simple lecteur de TMX qui transfère les informations des fichiers Tiled via une structure de classe faite à la main, mais il ne charge pas les données des images.
- Spécifique aux Androids :
 - `AndroidTMXLoader <https://github.com/davidmi/Android-TMX-Loader>` charge des données TMX dans un objet et génère un rendu pour une Bitmap Android (fonctionnalités limitées)
 - `libtiled-java port` est une déclinaison de la bibliothèque libtiled-java utilisée pour les téléphones Android.

18.1.9 PHP

- `PHP TMX Viewer` par sebbu : rend la carte en tant qu'image (autorise aussi quelques modifications)

18.1.10 Pike

- `TMX parser` : un simple chargeur de carte TMX (uniquement en format CSV).

18.1.11 Processing

- `linux-man/ptmx` : Ajout de cartes Tiled pour vos dessins Processing.

18.1.12 Python

- `pytiled-parser` : Analyseur Python pour les cartes en JSON (Pas de support pour le format TMX dans la version v1.0.0)
- `Arcade` : Bibliothèque de jeu 2D qui utilise pytiled-parser pour facilement charger des cartes JSON dans un jeu. (Pas de support pour le format TMX dans la version v2.6.0) [Exemples de Tiled dans Arcade](#)
- `pytmxlib` : bibliothèque pour des manipulations programmées de cartes TMX
- `python-tm` : une bibliothèque simple pour lire et écrire des fichiers TMX.

18.1.13 Ruby

- `tmx gem` par erisdiscord

18.1.14 Vala

- [librpg](#) Une bibliothèque pour charger et gérer des collection d'images (dans son propre format) et des cartes TMX orthogonales.

18.2 Support par Environnement

Les entrées suivantes sont des solutions intégrées pour des moteurs de jeux spécifiques. En gros, cela n'a pas ou peu d'importance si vous n'utilisez pas ces moteurs de jeu.

18.2.1 AndEngine

- [AndEngine](#) par Nicolas Gramlich supporte les rendus de cartes TMX

18.2.2 Allegro

- [allegro_tiled](#) intègre un support Tiled avec [Allegro 5](#).

18.2.3 Castle Game Engine (Pascal Objet)

- [Castle Game Engine](#) a un support pour des cartes Tiled (voir la section [CastleTiledMap](#))

18.2.4 Cell2D

- La bibliothèque Java [Cell2D](#) supporte les cartes Tiled via un accès à [TiledReader](#), mais a couramment un meilleur support intégré pour les cartes orthogonales que les autres types d'orientation.

18.2.5 cocos2d

- [cocos2d \(Python\)](#) supporte le chargement de [cartes Tiled](#) à travers le module `cocos.tiles`.
- [cocos2d-x \(C++\)](#) supporte le chargement de cartes TMX à travers la classe `CCTMXTiledMap`.
- [cocos2d-objc \(Objective-C, Swift\)](#) (précédemment connu comme : [cocos2d-iphone](#), [cocos2d-swift](#), [cocos2d-spritebuilder](#)) supporte le chargement de cartes TMX à travers `CCTiledMap`
- [TilemapKit](#) est un système de placement de tuiles pour Cocos2D. Il supporte tous les types de cartes de tuiles TMX, incluant les cartes isométriques, hexagonales et ses variations échelonnées. N'est plus en développement.

18.2.6 Construct 2 - Scirra

- [Construct 2](#), supporte officiellement les cartes TMX et l'import par un simple dépôt du fichier dans l'éditeur depuis la version Beta 149. [Note Officielle](#)

18.2.7 Flixel

- Lithander a démontré son analyseur Flash TMX combiné avec du rendu Flixel

18.2.8 Game Maker

- Tiled est fourni avec un greffon qui peut exporter une carte vers un fichier de salle pour GameMaker : Studio 1.4.
- [Tiled2GM Converter](#) par Dmi7ry

18.2.9 Godot

- [Tiled Map Importer](#) importe chaque carte en tant que scène Godot qui peut être instanciée ou héritée (annonce du forum).
- [Tiled To Godot Export](#) est une *extension JavaScript* pour Tiled qui exporte des cartes de tuiles et des jeux de tuiles dans le format Godot 3.2 (annonce du forum).

18.2.10 Grid Engine

- Grid Engine de Planimeter supporte les cartes Tiled exportées en Lua.

18.2.11 Haxe

- [HaxePunk](#) Chargeur Tiled pour HaxePunk
- [HaxeFlixel](#)
- OpenFL « openfl-tiled » est une bibliothèque qui donne la possibilité d'utiliser l'Éditeur de Carte Tiled aux développeurs OpenFL.
- [OpenFL + Tiled + Flixel](#) Colle expérimentale pour utiliser « openfl-tiled » avec HaxeFlixel

18.2.12 HTML5 (plusieurs moteurs)

- [Canvas Engine](#) Un environnement pour créer des jeux vidéos dans une Canvas HTML5
- [chem-tmx](#) Greffon pour le moteur de jeu [chem](#).
- [chesterGL](#) Une simple bibliothèque de jeu WebGL/canvas
- [Crafty](#) Moteur de Jeu en JavaScript HTML5 ; supporte le chargement de cartes Tiled à travers le composant externe [TiledMapBuilder](#).
- [GameJs](#) Une bibliothèque en JavaScript pour de la programmation de jeu vidéo ; un encapsulateur simple pour dessiner sur une toile HTML5 et d'autres modules utiles pour le développement de jeu vidéo
- [KineticJs-Ext](#) Une bibliothèque pour jeux ayant un rendu multi-toile
- [melonJS](#) Un moteur de jeu HTML5 léger
- [Panda 2](#), une Plateforme de Développement de Jeu en HTML5 pour Mac, Windows et Linux. A un greffon pour afficher des cartes Tiled, qu'elles soient orthogonales ou isométriques.
- [Phaser](#) Un environnement libre rapide, gratuit et amusant qui supporte le JavaScript et le TypeScript (tutoriel Tiled)
- [linux-man/p5.tiledmap](#) ajoute des cartes Tiled à p5.js.
- [Platypus Engine](#) Un moteur de tuiles orthogonales robuste avec une bibliothèque d'entités de jeu.
- [sprite.js](#) Un système de jeu pour des images.
- [TMXjs](#) Un analyseur et rendeur de TMX (XML de Carte de Tuile) basé sur du JavaScript, jQuery et RequireJS.
- [glazeJS](#) Un moteur de jeu 2D à hautes performances créé avec Typescript. Il supporte le format TMX, et crée un rendu de calques de tuiles sur le GPU via WebGL ([démo](#)).

18.2.13 **indielib-crossplatform**

- [indielib cross-platform](#) supporte le chargement de cartes TMX à travers [tmx-parser basé sur C++/TinyXML](#) par KonoM (BSD)

18.2.14 **LibGDX**

- [libgdx](#), une bibliothèque de jeu Android/PC/HTML5 basée sur du Java, [fournit](#) un emballeur, chargeur et rendeur de cartes TMX

18.2.15 **LITIengine**

- [LITIengine](#) est un Moteur de Jeu 2D en Java qui supporte le chargement, la sauvegarde et le rendu de cartes dans le format .tmx.

18.2.16 **LÖVE**

- [Simple Tiled Implementation](#) chargeur Lua pour le moteur de jeu LÖVE (Love2d).

18.2.17 **MOAI SDK**

- [Hanappe](#) Environnement pour le SDK MOAI.
- [Rapanui](#) Environnement pour le SDK MOAI.

18.2.18 **Monkey X**

- [bit.tiled](#) Charge des fichiers TMX en tant qu'objets. Vise la compatibilité complète avec des fichiers TMX natifs.
- [Diddy](#) est un environnement extensif pour Monkey X qui contient un module de chargement et de génération de rendu de fichier TMX. Supporte les cartes orthogonales et isométriques en tant que CSV et Base64 (non compressé).

18.2.19 **Node.js**

- [node-tmx-parser](#) - charge le fichier TMX dans un objet JavaScript

18.2.20 **Oak Nut Engine (onut)**

- [Oak Nut Engine](#) supporte les cartes Tiled à travers du Javascript et du C++. (voir des échantillons de TiledMap en Javascript ou en C++)

18.2.21 Orx Portable Game Engine

- TMX to ORX Converter Téléchargement du tutoriel et du convertisseur pour Orx.

18.2.22 Pygame

- Pygame map loader par dr0id
- PyTMX par Leif Theden (bitcraft)
- tmx.py par Richard Jones, de sa présentation sur “Introduction au Développement de Jeu” à la PyCon 2012.
- TMX, une branche de tmx.py et un portage sur Python3. Une démonstration nommée pylletTown peut être trouvée [ici](#).

18.2.23 Pyglet

- Chargeur/rendeur de carte JSON pour pyglet par Juan J. Martínez (reidrac)
- PyTMX par Leif Theden (bitcraft)

18.2.24 PySDL2

- PyTMX par Leif Theden (bitcraft)

18.2.25 RPG Maker MV

- Greffon Tiled pour RPG Maker MV par Dr.Yami & Archeia, de [RPG Maker Web](#)

18.2.26 SDL

- Chargeur basé sur du C++/TinyXML/SDL exemple par Rohin Knight (fonctionnalités limitées)

18.2.27 SFML

- STP (Analyseur de TMX pour SFML) par edoren
- Chargeur de carte Tiled pour du C++/SFML par fallahn. (Zlib/libpng)
- C++/SfTileEngine par Tresky (fonctionnalités limitées pour le moment)

18.2.28 Slick2D

- Slick2D supporte le chargement de cartes TMX à travers [TiledMap](#).

18.2.29 Solar2D (avant connu sous le nom de Corona SDK)

- [ponytiled](#) est un simple Chargeur de Cartes Tiled pour Solar2D ([annonce sur le forum](#))
- [Dusk Engine](#) est un moteur de cartes Tiled plein de fonctionnalités pour solar2D (n'est plus maintenu, mais peut quand même être utile)
- [Berry](#) est un simple Chargeur de Cartes Tiled pour Solar2D.
- [Qiso](#) est un moteur isométrique pour Solar2D qui supporte le chargement de cartes Tiled, et peut aussi gérer des choses telles que la recherche de chemin pour vous.

18.2.30 Sprite Kit Framework

- [SKTilemap](#) a été créé à partir de rien en Swift. Il est mis à jour, rempli de fonctionnalités et facile à intégrer dans n'importe quel projet Sprite Kit. Supporte l'iOS et OSX.
- [SKTiled](#) - Un système en Swift pour travailler avec des ressources Tiled dans SpriteKit.
- [JSTileMap](#) est une implémentation légère du format TMX pour SpriteKit supportant l'iOS 7 et l'OS X 10.9 et ultérieur.

18.2.31 TERRA Engine (Delphi/Pascal)

- [TERRA Engine](#) supporte le chargement et le rendu des cartes TMX.

18.2.32 Unity

- [SuperTiled2Unity](#) est une collection de scripts C# Unity qui peut automatiquement importer des fichiers de cartes de Tiled directement dans vos projets Unity.
- [Tiled TMX Importer](#), qui importe dans le nouveau système de Jeu de Tuiles natif d'Unity 2017.2.
- [Tiled To Unity](#) est un pipeline 3D pour des cartes Tiled. Il utilise des préfabs en tant que tuiles, et peut placer des décorations sur des tuiles dynamiquement. Supporte plusieurs calques (incluant les calques d'objets).
- [Tuesday](#) : Un sérialiseur et déserialiseur ainsi qu'une collection de scripts d'éditeur Unity qui vous permet de glisser-poser des fichiers TMX dans votre scène, les éditer, et sauvegarder les changements en tant que fichiers TMX. Licence MIT.
- [UniTiled](#), un importeur TMX natif pour Unity.
- [X-UniTmx](#) supporte presque toutes les fonctionnalités de Tiled 0.11. Importe des fichiers TMX/XML en tant que des Objets d'Image ou des Filets.
- [Orthello Pro](#) (Système 2D) offre un [support pour les cartes Tiled](#).

18.2.33 Unreal Engine 4

- [Paper2D](#) fournit un support intégré pour des cartes de tuiles et des jeux de tuiles, en important du JSON exporté depuis Tiled.

18.2.34 Urho3D

- Urho3D supporte le chargement de cartes Tiled nativement en tant que partie de la sous-bibliothèque Urho2D (Documentation, Exemple en HTML5).

18.2.35 XNA

- FlatRedBall L'outil de glue est distribué avec un greffon Tiled qui charge les cartes TMX dans le moteur FlatRedBall, qui fournit une intégration riche avec ses fonctionnalités.
- XTiled par Michael C. Neel et Dylan Wolf, une bibliothèque XNA qui permet de charger et de générer un rendu de cartes TMX
- XNA map loader par Kevin Gadd, étendu par Stephen Belanger et Zach Musgrave

CHAPITRE 19

Format de Carte TMX

Version 1.5

Les formats TMX et TSX sont les formats utilisés par [Tiled](#) pour stocker les cartes de tuiles et les jeux de tuiles, basés sur le XML. Le format TMX fournit un moyen flexible de décrire une carte basée sur des tuiles. Il peut décrire des cartes avec n'importe quelle taille de tuile, tout nombre de calques, tout nombre de collection de tuiles et il autorise l'ajout de propriétés personnalisées pour la majorité des éléments. Mis à part les calques de tuiles, il peut aussi contenir des groupes d'objets qui peuvent être placés librement.

Veuillez noter qu'il y a beaucoup de [*bibliothèques et d'environnements*](#) disponibles qui supportent les cartes TMX et les jeux de tuiles TSX.

Dans ce document, nous allons énumérer tous les éléments inclus dans ces formats de fichier. Les éléments sont mentionnés dans les en-têtes et la liste des attributs de ces éléments sont listés ci-dessous, suivi d'une courte explication. Les attributs et les éléments qui sont obsolètes ou non supportés par la version courante de Tiled sont en italique. Tous les attributs optionnels sont soit marqués en tant qu'optionnels, soit ils ont une valeur par défaut qui implique le fait qu'ils sont optionnels.

Veuillez regarder les [*notes de versions*](#) si vous êtes intéressés par les changements entre les différentes versions de Tiled.

Note : Un fichier DTD (Définition de Type de Document) est fourni dans <http://mapeditor.org/dtd/1.0/map.dtd>. Ce fichier n'est pas à jour mais peut être utile pour les espaces de noms en XML.

Note : Pour des raisons de compatibilité, il est recommandé d'ignorer les éléments et attributs inconnus (ou d'afficher un avertissement). Cela rend l'ajout de fonctionnalités plus facile sans casser la rétrocompatibilité, et permet des variations personnalisées et des additions avec lesquelles travailler avec les outils existants.

19.1 <map>

- **version** : La version du format TMX. « 1.0 » jusqu'à présent, et sera incrémenté pour refléter les sorties mineures de Tiled.
- **tiledversion** : La version de Tiled utilisée lors de la sauvegarde du fichier (depuis Tiled 1.0.1). Peut être une date (pour les versions instantanées). (optionnel)
- **orientation** : L'orientation de la carte. Tiled supporte « orthogonal », « isometric », « staggered » (échelonné) et « hexagonal » (depuis la 0.11).
- **renderorder** : L'ordre dans lequel les tuiles d'un calque de tuiles sont rendus. Les valeurs valides sont `right-down` (en bas à droite, par défaut), `right-up` (en haut à droite), `left-down` (en bas à gauche) et `left-up` (en haut à droite). Dans tous les cas, la carte est dessinée ligne par ligne. (seulement supporté pour les cartes orthogonales pour le moment)
- **compressionlevel** : Le niveau de compression à utiliser pour les données d'un calque de tuiles (-1 par défaut, ce qui veut dire que l'algorithme par défaut est utilisé).
- **width** : La longueur de la carte en tuiles.
- **height** : La hauteur de la carte en tuiles.
- **tilewidth** : La longueur d'une tuile.
- **tileheight** : La hauteur d'une tuile.
- **hexsidelength** : Seulement pour les cartes hexagonales. Détermine la longueur ou hauteur (dépendant de l'axe d'échelonnage) du côté de la tuile, en pixels.
- **staggeraxis** : Pour les cartes échelonnées ou hexagonales, détermine quel axe (« x » or « y ») est échelonné. (depuis la 0.11)
- **staggerindex** : Pour les cartes échelonnées ou hexagonales, détermine si les index « pairs » ou « impairs » sur l'axe échelonné sont décalés. (depuis la 0.11)
- **backgroundcolor** : La couleur d'arrière-plan de la carte. (optionnel, peut inclure une valeur de transparence depuis la 0.15 de la forme #AARRVVBB. Complètement transparent par défaut.)
- **nextlayerid** : Stocke le prochain ID disponible pour de nouveaux calques. Ce nombre est stocké pour empêcher la réutilisation du même ID après que des calques ont été supprimés. (depuis la 1.2) (le plus grand id de calque dans le fichier + 1 par défaut)
- **nextobjectid** : Stocke le prochain ID disponible pour de nouveaux objets. Ce nombre est stocké pour empêcher la réutilisation du même ID après que des objets ont été supprimés. (depuis la 0.11) (le plus grand id d'objet dans le fichier + 1 par défaut)
- **infinite** : Si la carte est infinie. Une carte infinie n'a pas de taille fixe et peut grandir dans toutes les directions. Ses données de calques sont stockées en tant qu'échantillons. (0 pour faux, 1 pour vrai, 0 par défaut)

Les propriétés `tilewidth` et `tileheight` déterminent la taille générale de la grille de la carte. Les tuiles individuelles peuvent avoir des tailles différentes. Les plus grandes tuiles dépasseront vers le haut et la droite (ancré en bas à gauche).

Une carte contient trois différents types de calques. Les calques de tuiles étaient auparavant le seul type de calque, et étaient simplement appelés `layer`, les calques d'objets ont la balise `objectgroup` et les calques d'images utilisent la balise `imagelayer`. L'ordre dans lequel ces calques apparaissent est l'ordre dans lequel ces calques sont rendus par Tiled.

L'orientation `staggered` réfère à une carte isométrique utilisant des axes échelonnés.

Can contain at most one : `<properties>`, `<editorsettings>` (since 1.3)

Can contain any number : `<tileset>`, `<layer>`, `<objectgroup>`, `<imagelayer>`, `<group>` (since 1.0)

19.2 <editorsettings>

Cet élément contient plusieurs options spécifiques à l'éditeur, qui sont généralement inutiles lors de la lecture d'une carte.

Can contain at most one : *<chunksize>*, *<export>*

19.2.1 <chunksize>

- **width** : La longueur des échantillons utilisés pour les cartes infinies (16 par défaut).
- **height** : La hauteur des échantillons utilisés pour les cartes infinies (16 par défaut).

19.2.2 <export>

- **target** : Le dernier fichier dans lequel cette carte a été exportée.
- **format** : Le nom court du dernier format dans lequel cette carte a été exporté.

19.3 <tilesset>

- **firstgid** : Le premier ID de tuile global du jeu de tuiles (cet ID global correspond à la première tuile du jeu de tuiles).
- **source** : Si ce jeu de tuiles est stocké dans un fichier TSX (Jeu de Tuiles XML) externe, cet attribut réfère ce fichier. Ce fichier TSX a la même structure que l'élément <tilesset> décrit ici. (L'attribut firstgid est manquant et l'attribut source est aussi absent. Ces deux attributs sont gardés dans la carte TMX car ils sont spécifiques aux cartes.)
- **name** : Le nom de ce jeu de tuiles.
- **tilewidth** : La longueur (maximale) des tuiles de ce jeu de tuiles.
- **tileheight** : La hauteur (maximale) des tuiles de ce jeu de tuiles.
- **spacing** : L'espacement entre les tuiles de ce jeu de tuiles en pixels (s'applique à l'image du jeu de tuiles, 0 par défaut)
- **margin** : La marge autour des tuiles de ce jeu de tuiles (s'applique à l'image du jeu de tuiles, 0 par défaut)
- **tilecount** : Le nombre de tuiles de ce jeu de tuiles (depuis la 0.13)
- **columns** : Le nombre de colonnes de tuiles de ce jeu de tuiles. Modifiable pour les jeux de tuiles de collection d'images, et est utilisé lors de l'affichage du jeu de tuiles. (depuis la 0.15)
- **objectalignment** : Contrôle l'alignement des objets tuiles. Les valeurs valides sont **unspecified** (non spécifié), **topleft** (en haut à gauche), **top** (en haut), **topright** (en haut à droite), **left** (à gauche), **center** (au centre), **right** (à droite), **bottomleft** (en bas à gauche), **bottom** (en bas) et **bottomright** (en bas à droite). La valeur par défaut est **unspecified** pour des raisons de compatibilité. Lorsque c'est non spécifié, les objets tuiles utilisent **bottomleft** en mode orthogonal et **bottom** en mode isométrique. (depuis la 1.4)

S'il y a plusieurs éléments <tilesset>, ils sont en ordre ascendant de leur attribut **firstgid**. Le premier jeu de tuiles a toujours une valeur de **firstgid** à 1. Depuis Tiled 0.15, les jeux de tuiles de collection d'images ne numérotent pas forcément leur tuiles de façon consécutives car des trous peuvent apparaître lors de la suppression de tuiles.

Les jeux de tuiles de collection n'ont pas de tag <image>. À la place, chaque tuile a un tag <image>.

Peut contenir au plus un : *image*, *décalage de jeu de tuiles*, *grille* (depuis la 1.0), *propriétés*, *types de terrains*, *collections Wang* (depuis la 1.1), *transformations de jeu de tuiles* (depuis la 1.5)

Peut contenir tout nombre de : *tuile de jeu de tuiles*

19.3.1 <tileoffset>

- **x** : Décalage horizontal en pixels. (0 par défaut)
- **y** : Décalage vertical en pixels (positif en bas, 0 par défaut)

Cet élément est utilisé pour spécifier un décalage en pixels à appliquer lors du dessin d'une tuile de ce jeu de tuiles. Aucun décalage n'est appliqué si cet élément n'est pas présent.

19.3.2 <grid>

- **orientation** : Orientation de la grille pour les tuiles de ce jeu de tuiles (`orthogonal` (orthogonale) ou `isometric` (isométrique), `orthogonal` par défaut)
- **width** : Longueur d'une cellule de la grille
- **height** : Hauteur d'une cellule de la grille

Cet élément est seulement utilisé pour une orientation isométrique, et détermine la façon dont les informations de terrain et de collision des tuiles sont rendues.

19.3.3 <image>

- **format** : Utilisé pour des images embarquées combiné avec l'élément fils `data`. Les valeurs valides sont des extensions de fichier tel que `png`, `gif`, `jpg`, `bmp`, etc.
- **id** : Utilisé par quelques versions de Tiled en Java. Obsolète et non supporté.
- **source** : La référence du fichier d'image du jeu de tuiles (Tiled supporte la majorité des formats d'image communs). Seulement utilisé si l'image n'est pas intégrée.
- **trans** : Définit une couleur spécifique traitée comme transparence (exemple : « #FF00FF » pour du magenta). L'ajout du `#` est optionnel et Tiled l'ignore pour des raisons de compatibilité. (optionnel)
- **width** : La longueur de l'image en pixels (optionnel, utilisé pour une correction d'index de tuiles quand l'image change)
- **height** : La hauteur de l'image en pixels (optionnel)

Veuillez noter qu'il n'est pour l'instant pas possible d'utiliser Tiled pour créer des cartes avec des données d'images embarquées même si le format TMX le supporte. Il est possible de créer de telles cartes en utilisant `libtiled`` (Qt/C++) ou `tmxlib` (Python).

Peut contenir au plus : *données*

19.3.4 <terraintypes>

Cet élément définit un tableau de types de terrains qui peut être référencé depuis l'attribut `terrain` ou l'élément `tile`.

Peut contenir tout nombre de : *terrain*

<terrain>

- **name** : Le nom du type de terrain.
- **tile** : L'id de tuile local de la tuile qui représente le terrain visuellement.

Peut contenir au plus un : *propriété*

19.3.5 <transformations>

Cet élément est utilisé pour décrire quelles transformations peuvent être appliquées aux tuiles (pour par exemple étendre une collection Wang en transformant les tuiles existantes).

- **hflip** : Si les tuiles de ce jeu de tuiles peuvent être inversées horizontalement (0 par défaut)
- **vflip** : Si les tuiles de ce jeu de tuiles peuvent être inversées verticalement (0 par défaut)
- **rotation** : Si les tuiles de ce jeu de tuiles peuvent être pivotées en utilisant des incrément de 90 degrés (0 par défaut)
- **preferuntransformed** : Si les tuiles non transformées sont préférées, sinon les tuiles transformées sont utilisées pour produire plus de variations (0 par défaut)

19.3.6 <tile>

- **id** : L'ID local de la tuile dans son jeu de tuiles.
- **type** : Le type de la tuile. Réfère à un type d'objet et est utilisé par des objets tuiles. (optionnel) (depuis la 1.0)
- **terrain** : Définit le type de terrain de chaque coin de la tuile, donné en tant qu'index appartenant au tableau de types de terrain séparés par des virgules dans l'ordre en haut à gauche, en haut à droite, en bas à gauche, en bas à droite. Une valeur oubliée veut dire que ce coin n'a aucun terrain. (optionnel)
- **probability** : Un pourcentage indiquant la probabilité que cette tuile soit choisie quand elle est en compétition avec d'autres tuiles lors de l'utilisation de l'outil de terrain. (0 par défaut)

Peut contenir au plus un : *propriétés*, image <tmx-image> (depuis la 0.9), *groupe d'objets*, *animation*

<animation>

Contient une liste de trames d'animation.

Chaque tuile peut avoir exactement une animation qui lui est attribuée. Dans le futur, il pourrait y avoir un support pour plusieurs animations nommées pour une tuile.

Peut contenir tout nombre de : *trame*

<frame>

- **tileid** : L'ID local d'une tuile appartenant au *jeu de tuiles* parent.
- **duration** : La longueur (en millisecondes) pendant laquelle cette trame doit être affichée avant d'afficher la prochaine trame.

19.3.7 <wangsets>

Contient la liste de collections Wang définies pour ce jeu de tuiles.

Peut contenir tout nombre de : *collection Wang*

<wangset>

Définit une liste de couleurs de coin et une liste de couleurs de bord, et n'importe quel nombre de tuiles Wang qui utilisent ces couleurs.

- **name** : Le nom de l'ensemble Wang.
- **tile** : L'ID de tuile de la tuile qui représente cet ensemble Wang.

Peut contenir au plus un : *propriété*

Peut contenir jusqu'à 255 : *couleur Wang* (depuis Tiled 1.5)

Peut contenir tout nombre de : *tuile Wang*

<wangcolor>

Une couleur utilisable pour définir le coin et/ou bord d'une tuile Wang.

- **name** : Le nom de cette couleur.
- **color** : La couleur dans le format #RRVVBB (exemple : #c17d11).
- **tile** : L'ID de tuile de la tuile représentant cette couleur.
- **probability** : La probabilité relative que cette couleur soit choisie parmi d'autres s'il y a plusieurs options. (0 par défaut)

Peut contenir au plus un : *propriété*

<wangtile>

Définit une tuile Wang en référant une tuile du jeu de tuiles et en l'associant à un ID Wang.

- **tileid** : L'ID de la tuile.
- **wangid** : « L'ID Wang est donné par une liste d'index séparés par des virgules (en commençant par 1, car 0 veut dire _non associé_) référant à des couleurs Wang dans la collection Wang dans l'ordre suivant : en haut, en haut à droite, à droite, en bas à droite, en bas, en bas à gauche, à gauche, en haut à gauche (depuis Tiled 1.5). Avant Tiled 1.5, l'ID Wang était sauvegardé en tant qu'entier non-signé de 32-bits stocké dans le format 0xCECECECE (où chaque C est une couleur de coin et chaque E une couleur de bord, dans l'ordre inverse). »
- **hflip** : Si la tuile peut être inversée horizontalement (enlevé dans Tiled 1.5).
- **vflip** : Si la tuile peut être inversée verticalement (enlevé dans Tiled 1.5).
- **dflip** : Si la tuile peut être inversée diagonalement (enlevé dans Tiled 1.5).

19.4 <layer>

Toutes les balises *jeu de tuiles* doivent être avant la première balise *calque* pour que les analyseurs puissent avoir les jeux de tuiles avant de résoudre les tuiles.

- **id** : ID unique de ce calque. Chaque calque qui est ajouté dans une carte obtient un id unique. Même si un objet a été supprimé, aucun objet ne reçoit le même ID. Ne peut être changé dans Tiled. (depuis Tiled 1.2)
- **name** : Le nom de ce calque. (» » par défaut)
- **x** : La coordonnée X du calque en tuiles. 0 par défaut et ne peut pas être modifié dans Tiled.
- **y** : La coordonnée Y du calque en tuiles. 0 par défaut et ne peut pas être modifié dans Tiled.
- **width** : La longueur du calque en tuiles. Toujours la même que la carte pour les cartes à taille fixe.
- **height** : La hauteur du calque en tuiles. Toujours la même que la carte pour les cartes à taille fixe.
- **opacity** : L'opacité du calque en tant que valeur entre 0 et 1. 1 par défaut.
- **visible** : Détermine si le calque est visible (1) ou caché (0). 1 par défaut.
- **tintcolor** : Une *couleur de teinte* qui est multipliée par toutes les tuiles dessinées dans ce calque dans le format #AARRVVBB ou #RRVVBB (optionnel).
- **offsetx** : Décalage horizontal de ce calque en pixels. 0 par défaut. (depuis la 0.14)

- **offsety** : Décalage vertical de ce calque en pixels. 0 par défaut. (depuis la 0.14)
- **parallax** : *Facteur de parallaxe* horizontal pour ce calque. 1 par défaut. (depuis la 1.5)
- **parallaxy** : *Facteur de parallaxe* vertical pour ce calque. 1 par défaut. (depuis la 1.5)

Peut contenir au plus un : *propriétés, données*

19.4.1 <data>

- **encoding** : L'encodage utilisé pour encoder les données du calque de tuiles. Peut être « base64 » ou « csv » pour le moment s'il est utilisé. (optionnel)
- **compression** : L'algorithme de compression utilisée pour compresser les données du calque de tuiles. Tiled supporte « gzip » et « zlib » et (en tant qu'option lors de la compilation depuis Tiled 1.3) « ztsd ».

Quand aucun encodage ou algorithme de compression est donné, les tuiles sont stockées en tant qu'éléments XML `tile` individuels. De plus, le format le plus facile à analyser est le format « csv » (valeurs séparées par des virgules).

Les données de calque encodées en base64 et celles qui sont optionnellement compressées sont plus difficiles à analyser. Tout d'abord, vous devez utiliser un décodeur de base64, puis vous pouvez avoir besoin de le décompresser. Maintenant que vous avez un tableau d'octets, qui doit être interprété comme une liste d'entiers non signés de 32-bits en utilisant l'ordre d'octets little endian.

Whatever format you choose for your layer data, you will always end up with so called « *Global Tile IDs* » (gids). They are called « global », since they may refer to a tile from any of the tilessets used by the map. The IDs also contain *flipping flags*. The tilessets are always stored with increasing *firstgid*.

Peut contenir tout nombre de : *tuile de calque de tuiles, échantillon*

19.4.2 <chunk>

- **x** : La coordonnée X de l'échantillon en tuiles.
- **y** : La coordonnée Y de l'échantillon en tuiles.
- **width** : La longueur de l'échantillon en tuiles.
- **height** : La hauteur de l'échantillon en tuiles.

Cette fonctionnalité est pour le moment seulement implémentée pour les cartes infinies. Le contenu d'un élément échantillon est le même que celui de l'élément `data`, sauf qu'il stocke les données de l'aire spécifié dans ses attributs.

Peut contenir tout nombre de : *tuile de calque de tuiles*

19.4.3 <tile>

- **gid** : L'ID de tuile global (0 par défaut).

À ne pas confondre avec l'élément `tile` dans un `tilesset`, cet élément définit la valeur d'une tuile simple dans un calque de tuiles. C'est cependant le moyen le plus inefficace de stocker les données de calque de tuiles, et doit être évité de façon générale.

19.5 <objectgroup>

- **id** : ID unique de ce calque. Chaque calque qui est ajouté dans une carte obtient un id unique. Même si un objet a été supprimé, aucun objet ne reçoit le même ID. Ne peut être changé dans Tiled. (depuis Tiled 1.2)
- **name** : Le nom du groupe d'objets. (» » par défaut)
- **color** : La couleur utilisée pour afficher les objets de ce groupe. (gris (« #a0a0a4 ») par défaut)
- **x** : La coordonnée X de ce groupe d'objets en tuiles. 0 par défaut et ne peut plus être changé dans Tiled.
- **y** : La coordonnée Y de ce groupe d'objets en tuiles. 0 par défaut et ne peut plus être changé dans Tiled.
- **width** : La longueur du groupe d'objets en tuiles. Vide de sens.
- **height** : La hauteur du groupe d'objets en tuiles. Vide de sens.
- **opacity** : L'opacité du calque en tant que valeur entre 0 et 1. (1 par défaut)
- **visible** : Détermine si le calque est visible (1) ou caché (0). (1 par défaut)
- **tintcolor** : Une couleur qui est multipliée par tous les objets tuiles dessinés dans ce calque, dans le format #AARRVVBB ou #RRVVBB (optionnel).
- **offsetx** : Décalage horizontal de ce groupe d'objets en pixels. (0 par défaut) (depuis la 0.14)
- **offsety** : Décalage vertical de ce groupe d'objets en pixels. (0 par défaut) (depuis la 0.14)
- **draworder** : Détermine si les objets sont rendus en suivant l'ordre d'apparition (« index ») ou en suivant leur coordonnée Y (« topdown »). (« topdown » par défaut)

Le groupe d'objets est en fait un calque de carte, et est par conséquent appelé « calque d'objets » dans Tiled.

Peut contenir au plus un : *propriété*

Peut contenir tout nombre de : *objet*

19.5.1 <object>

- **id** : ID unique de cet objet. Chaque objet qui est placé dans une carte obtient un id unique. Même si un objet a été supprimé, aucun objet ne reçoit le même ID. Ne peut être changé dans Tiled. (depuis Tiled 0.11)
- **name** : Le nom de cet objet. Une chaîne de caractères arbitraire. (» » par défaut)
- **type** : Le type de cet objet. Une chaîne de caractères arbitraire. (» » par défaut)
- **x** : La coordonnée X de cet objet en pixels. (0 par défaut)
- **y** : La coordonnée Y de cet objet en pixels. (0 par défaut)
- **width** : La longueur de cet objet en pixels. (0 par défaut)
- **height** : La hauteur de cet objet en pixels. (0 par défaut)
- **rotation** : La rotation de cet objet en degrés dans le sens des aiguilles d'une montre autour de (x, y). (0 par défaut)
- **gid** : Une référence vers une tuile. (optionnel)
- **visible** : Détermine si l'objet est visible (1) ou caché (0). (1 par défaut)
- **template** : Une référence vers un *fichier de modèle*. (optionnel)

Même si les calques de tuiles sont très bons pour quelque chose de répétitif qui est aligné sur la grille de tuiles, parfois il est utile d'annoter votre carte avec d'autres informations qui ne sont nécessairement pas alignées sur la grille. Voici pourquoi les objets ont leurs coordonnées et taille en pixels, mais vous pouvez aussi facilement les aligner sur la grille quand vous le voulez.

Les objets sont souvent utilisés pour ajouter des informations personnalisées à votre carte de tuiles tel que des points d'apparition, des téléporteurs, des sorties, etc.

Quand l'objet a un attribut **gid**, alors il est représenté par l'image de la tuile ayant cet ID global. L'alignement de l'image correspond couramment à l'orientation de la carte. Il est aligné en bas à gauche pour une direction orthogonale tandis qu'il est aligné en bas au centre pour une direction isométrique. L'image va être pivotée respectivement autour d'en bas à gauche ou d'en bas au centre.

Quand l'objet a un attribut **template**, il va emprunter toutes les propriétés du modèle spécifié, avec les propriétés sauvegardées dans l'objet ayant la priorité la plus haute, ce qui veut dire qu'elles remplaceront les propriétés du modèle.

Peut contenir au plus un : *propriétés*, *ellipse* (depuis la 0.9), *point* (depuis la 1.1), *polygone*, *polyligne*, *texte* (depuis la 1.0)

19.5.2 <ellipse>

Utilisé pour traiter un objet en tant qu'une ellipse. Les attributs existants `x`, `y`, `width` et `height` sont utilisés pour déterminer la taille de l'ellipse.

19.5.3 <point>

Utilisé pour traiter un objet en tant qu'un point. Les attributs existants `x` et `y` sont utilisés pour déterminer la position du point.

19.5.4 <polygon>

- **points** : Une liste de coordonnées x,y en pixels.

Chaque objet `polygon` est fait d'une liste de coordonnées x,y délimitées dans l'espace. L'origine de ces coordonnées est la location de l'`object` parent. Par défaut, le premier point est créé en tant que 0,0 ce qui veut dire que le l'origine du point est exactement là où l'`object` est placé.

19.5.5 <polyligne>

- **points** : Une liste de coordonnées x,y en pixels.

Une `polyligne` suit la même définition de placement qu'un objet `polygon`.

19.5.6 <text>

- **fontfamily** : La famille de police utilisée (« sans-serif » par défaut)
- **pixelsize** : La taille de la police en pixels (pas d'utilisation de points, car d'autres tailles dans le format TMX utilisent aussi des pixels) (16 par défaut)
- **wrap** : Détermine si l'ajustement de texte est activé (1) ou désactivé (0). (0 par défaut)
- **color** : Couleur du texte dans le format #AARRVVBB ou #RRVVBB (#000000 par défaut)
- **bold** : Détermine si le texte est en gras (1) ou non (0). (0 par défaut)
- **italic** : Détermine si le texte est en italique (1) ou non (0). (0 par défaut)
- **underline** : Détermine si une ligne doit être dessinée sous le texte (1) ou non (0). (0 par défaut)
- **strikeout** : Détermine si une ligne doit être dessinée à travers le texte (1) ou non (0). (0 par défaut)
- **kerning** : Détermine si le crénage doit être utilisé (1) lors du rendu du texte ou non (0). (1 par défaut)
- **halign** : Alignement horizontal du texte dans l'objet (`left` (gauche), `center` (centré), `right` (droite) ou `justify` (justifié), `left` par défaut) (depuis Tiled 1.2.1)
- **valign** : Alignement vertical du texte dans l'objet (`top` (haut), `center` (centré) ou `bottom` (bas), `top` par défaut)

Utilisé pour traiter un objet en tant qu'un objet de texte. Contient le texte utilisé en tant que données de caractères.

Pour des raisons d'alignement, le bas du texte est la hauteur descendante de la police, et le haut du texte est la hauteur ascendante de la police. Par exemple, l'alignement `bottom` du mot « chat » va laisser de la place sous le texte, même si cet espace n'est pas utilisé pour ce mot pour la plupart des polices. De la même façon, l'alignement `top` du mot « chat » va laisser un peu d'espace au dessus du « t » pour la plupart des polices, car cet espace est utilisé pour des diacritiques.

Si le texte est plus grand que les bords de l'objet, il est coupé aux bords de l'objet.

19.6 <imagelayer>

- **id** : ID unique de ce calque. Chaque calque qui est ajouté dans une carte obtient un id unique. Même si un objet a été supprimé, aucun objet ne reçoit le même ID. Ne peut être changé dans Tiled. (depuis Tiled 1.2)
- **name** : Le nom du calque d'images. (» » par défaut)
- **offsetx** : Décalage horizontal de ce calque d'images en pixels. (0 par défaut) (depuis la 0.15)
- **offsety** : Décalage vertical de ce calque d'images en pixels. (0 par défaut) (depuis la 0.15)
- **x** : La position X du calque d'images en pixels. (0 par défaut, obsolète depuis la 0.15)
- **y** : La position Y du calque d'images en pixels. (0 par défaut, obsolète depuis la 0.15)
- **opacity** : L'opacité du calque en tant que valeur entre 0 et 1. (1 par défaut)
- **visible** : Détermine si le calque est visible (1) ou caché (0). (1 par défaut)
- **tintcolor** : Une couleur qui est multipliée par l'image dessinée dans ce calque, dans le format #AARRVVBB ou #RRVVBB (optionnel).

Un calque contenant une seule image.

Peut contenir au moins un : *propriétés, image*

19.7 <group>

- **id** : ID unique de ce calque. Chaque calque qui est ajouté dans une carte obtient un id unique. Même si un objet a été supprimé, aucun objet ne reçoit le même ID. Ne peut être changé dans Tiled. (depuis Tiled 1.2)
- **name** : Le nom du calque de groupes. (» » par défaut)
- **offsetx** : Décalage horizontal de ce groupe de calques en pixels. (0 par défaut)
- **offsety** : Décalage vertical de ce groupe de calques en pixels. (0 par défaut)
- **opacity** : L'opacité du calque en tant que valeur entre 0 et 1. (1 par défaut)
- **visible** : Détermine si le calque est visible (1) ou caché (0). (1 par défaut)
- **tintcolor** : Une couleur qui est multipliée par tout graphisme dessinée dans tout calque enfant, dans le format #AARRVVBB ou #RRVVBB (optionnel).

Un groupe de calques est utilisé pour organiser les calques d'une carte dans une hiérarchie. Ses attributs **offsetx**, **offsety**, **opacity**, **visible** et **tintcolor** affectent ses calques fils récursivement.

Peut contenir au plus un : *propriété*

Peut contenir tout nombre de : *calque, groupe d'objets, calque d'images, groupe*

19.8 <properties>

Contient n'importe quel nombre de propriétés personnalisées. Peut être utilisé en tant que fils des éléments **map**, **tileset**, **tile** (quand il appartient à un **tileset**), **terrain**, **wangset**, **wangcolor**, **layer**, **objectgroup**, **object**, **imagelayer** et **group**.

Peut contenir tout nombre de : *propriété*

19.8.1 <property>

- **name** : Le nom de cette propriété.
- **type** : Le type de cette propriété. Peut être **string** (chaîne de caractères, par défaut), **int** (entier), **float** (flottant), **bool** (booléen), **color** (couleur), **file** (fichier) ou **object** (objet) (depuis la 0.16, avec les options **color** et **file** ajoutées dans la 0.17, et **object** ajoutée dans la 1.4).
- **value** : La valeur de cette propriété. (La chaîne de caractères par défaut est « « , le nombre par défaut est 0, le booléen par défaut est « false », la couleur par défaut est #00000000, le fichier par défaut est « . » (le dossier parent du fichier courant))

Les valeurs booléennes ont une valeur qui est soit « true » (vrai), soit « false » (faux).

Les propriétés de couleur sont stockés en utilisant le format #AARRVVBB.

Les propriétés de fichier sont stockés en tant que chemin relatif depuis la location du fichier de carte.

Les propriétés d'objets peuvent référencer n'importe quel objet dans la même carte et sont stockés en tant qu'entier (l'ID de l'objet référencé, ou 0 lorsqu'aucun objet n'est référencé). Lorsque des objets sont utilisés dans l'Éditeur de collision de Tuiles, elles peuvent seulement référencer d'autres objets sur la même tuile.

Quand une propriété de chaîne de caractères contient des retours à la ligne, la version courante de Tiled montrera la valeur en tant que caractères contenus dans l'élément **property** à la place d'être dans l'attribut **value**. Il est possible qu'une future version du format TMX fasse en sorte que les valeurs des propriétés soient sauvegardées dans l'élément plutôt qu'en tant qu'un attribut.

19.9 Fichiers de Modèle

Les modèles sont sauvegardés dans leur propre fichier, et sont référencés par des *objets* qui sont des instances de modèle.

19.9.1 <template>

L'élément racine du modèle contient *l'objet de carte* sauvegardé et un élément *jeu de tuiles* qui pointe à un jeu de tuiles externe, si l'objet est un objet tuile.

Exemple d'un fichier de modèle :

```
<?xml version="1.0" encoding="UTF-8"?>
<template>
  <tileset firstgid="1" source="desert.tsx"/>
  <object name="cactus" gid="31" width="81" height="101"/>
</template>
```

Peut contenir au plus un : *jeu de tuiles*, *objet*



Fig. 1 – Licence Creative Commons

Le **Format de Carte TMX** de <http://www.mapeditor.org> est licencié sous une Licence Creative Commons Attribution-ShareAlike 3.0 Unported.

CHAPITRE 20

Notes de Version de TMX

Les changements et additions qui ont été effectués au format *Format de Carte TMX* pour les versions récentes de Tiled sont décrits ci-dessous.

20.1 Tiled 1.7

- Les éléments *tuiles d'un jeu de tuiles* dans un jeu de tuiles ne sont plus sauvegardés avec un ID incrémental. Ils sont maintenant sauvegardé dans l'ordre d'affichage, ce qui peut être configuré dans Tiled.

20.2 Tiled 1.5

- Les couleurs qui font partie d'une *collection Wang* ne sont plus séparées en couleurs de coins et de contours. À la place, il y a maintenant un seul élément *couleur Wang* pour définir une couleur Wang. Ce nouvel élément stocke aussi les *propriétés*.
- L'attribut *wang_id* de l'élément *tuile Wang* est maintenant stocké en tant qu'une liste de valeurs séparées par des virgules, plutôt qu'un nombre en 32-bits non-signé dans un format hexadécimal. C'est parce que le nombre de couleurs supportées dans une collection Wang a été augmenté de 15 à 255.
- Les transformations valides des tuiles d'une collection (inversion, rotation) sont spécifiées dans un élément *transformations de jeu de tuiles*. Le support partiel des attributs *vflip*, *hflip* et *dflip* de l'élément *tuile Wang* a été enlevé.
- L'élément *collection Wang* a remplacé l'élément *type de terrain* qui est maintenant déprécié.

20.3 Tiled 1.4

- Ajout de l'attribut `objectalignment` à l'élément `jeu de tuiles`, qui permet au jeu de tuiles de contrôler l'alignement utilisé pour les objets tuiles.
- Ajout de l'attribut `tintcolor` aux éléments `calque`, `groupe d'objets`, `calque d'images` et `groupe`, ce qui permet d'ajouter de nombreux effets graphiques tel que l'assombrissement ou la coloration d'un calque.

20.4 Tiled 1.3

- Ajout d'un élément `options de l'éditeur`, qui est utilisé pour stocker des options spécifiques à l'éditeur qui sont généralement inutiles lors du chargement d'une carte.
- Ajout du support pour la compression en Zstandard pour les données d'un calque de tuiles (`compression="zstd"` pour les éléments `données`).
- Ajout de l'attribut `compressionlevel` à l'élément `carte`, qui stocke le niveau de compression à utiliser pour les données des calques de tuiles.

20.5 Tiled 1.2.1

- Les objets de texte peuvent maintenant avoir un alignement horizontal sauvegardé en tant que `justify`. Cette option existait dans l'UI auparavant mais elle n'était pas sauvegardée proprement.

20.6 Tiled 1.2

- Ajout de l'attribut `id` aux éléments `calque`, `groupe d'objets`, `calque d'images` et `groupe`, qui stocke un ID unique par carte pour cet objet.
- Ajout de l'attribut `nextlayerid` à l'élément `carte` qui stocke le prochain ID disponible pour les nouveaux calques. Ce nombre est stocké pour prévenir la réutilisation du même ID après que des calques aient été supprimés.

20.7 Tiled 1.1

- Ajout d'un attribut `map.infinite`, qui indique si la carte est considérée sans bords. Les données des calques de tuiles pour les cartes infinies sont stockées en tant qu'échantillons.
- Un nouvel élément `échantillon` a été ajouté pour les cartes infinies qui contiennent un contenu proche de `données`, mis à part le fait qu'il stocke les données de l'aire spécifiée par ses attributs `x`, `y`, `width` et `height`.
- Des `modèles` ont été ajoutés, un modèle est un `fichier externe` référencé par des objets d'instance de modèle :

```
<object id="3" template="diamond.tx" x="200" y="100"/>
```

- Les jeux de tuiles peuvent maintenant contenir des `Collections de Terrain`. Elles sont sauvegardées dans le nouvel élément `collection Wang`.
- Un nouvel élément enfant `point` a été ajouté à l'élément `objet`, qui annote les objets points. Les objets points n'ont ni taille ni orientation.

20.8 Tiled 1.0

- Un nouveau élément *groupe* a été ajouté, qui est un groupe de calques qui peut avoir d'autres calques en tant qu'éléments fils. Cela veut dire que les calques forment maintenant une hiérarchie.
- Ajout d'objets texte, identifié par un nouvel élément *texte* qui est utilisé en tant qu'enfant de l'élément *objet*.
- Ajout de l'attribut *tile.type* pour le support de *types de tuiles*.

20.9 Tiled 0.18

Aucun changement de format de fichier.

20.10 Tiled 0.17

- Ajout de *color* et *file* comme valeurs possibles pour l'attribut *property.type*.
- Ajout du support d'édition de chaînes de caractères de plusieurs lignes, qui sont écrites différemment.

20.11 Tiled 0.16

- L'élément *propriété* a gagné un attribut *type* qui stocke le type de la valeur. Les types couramment supportés sont *string* (par défaut), *int*, *float* et *bool*.

20.12 Tiled 0.15

- Les attributs *offsetx* et *offsety* sont maintenant utilisés pour les éléments *calque d'images*, ce qui remplace les attributs *x* et *y* utilisés précédemment. Ce changement a été effectué pour garder une consistance entre les différents types de calques.
- Les tuiles d'un jeu de tuiles de collection d'images ne sont plus certainement consécutives, car la suppression de tuiles de la collection ne changera plus l'ID des autres tuiles.
- Les formats de calques de tuiles en XML pur et en Gzip compressé sont devenus obsolètes car ils n'avaient aucun avantage sur les autres formats. Les formats de données de calque restants sont le CSV, le Base64 et le Zlib compressé.
- Ajout de l'attribut *columns* à l'élément *jeu de tuiles* qui spécifie le nombre de colonnes de tuiles dans le jeu de tuiles. Pour les jeux de tuiles de collection d'images, la valeur est éditable et est utilisée quand le jeu de tuiles est affiché.
- L'attribut *backgroundcolor* de l'élément *carte* supporte le format #AARRVVBB lorsque la valeur de son alpha diffère de 255. Auparavant, la valeur d'alpha était ignorée silencieusement.

20.13 Tiled 0.14

- Ajout des attributs optionnels `offsetx` et `offsety` aux éléments `layer` et `objectgroup`. Ceux-ci spécifient un décalage en pixels appliqué lors du rendu du calque. Les valeurs par défaut sont 0.

20.14 Tiled 0.13

- Ajout d'un attribut optionnel `tilecount` à l'élément `tileset` qui est écrit par Tiled afin d'aider des analyseurs à déterminer l'espace mémoire à allouer pour les données des tuiles.

20.15 Tiled 0.12

- Les objets de tuiles n'avaient auparavant pas de propriétés `width` et `height`, même si le format l'acceptait techniquement. Maintenant, ces propriétés sont utilisées pour stocker la taille dans laquelle l'image doit être rendue. Les valeurs par défaut de ces attributs sont les dimensions de l'image de tuile.

20.16 Tiled 0.11

- Ajout d'`hexagonal` à la liste des valeurs supportées par l'attribut `orientation` de l'élément `map`. Cela ajoute aussi les attributs `staggerindex` (even ou odd), `staggeraxis` (x ou y) et `hexsidelength` (valeur entière) à l'élément `map` pour supporter les nombreuses variations de l'hexagonal échelonné. Les nouveaux attributs `staggerindex` et `staggeraxis` sont aussi supportés lors de l'utilisation de l'orientation de carte `staggered`.
- Ajout de l'attribut `id` aux éléments `object` qui stocke un ID unique par carte pour cet objet.
- Ajout de l'attribut `nextobjectid` à l'élément `map` qui stocke le prochain ID disponible pour de nouveaux objets. Ce nombre est stocké pour prévenir la réutilisation du même ID après que des objets aie été supprimés.

20.17 Tiled 0.10

- Les objets de tuiles peuvent maintenant être inversées horizontalement et verticalement. Ceci est stocké dans l'attribut `gid` en utilisant le même mécanisme que les tuiles normales. L'image est supposée être inversée sans affecter sa position, comme les tuiles inversées.
- Les objets peuvent être pivotés librement. La rotation est stockée en degrés dans l'attribut `rotation`, avec une valeur positive indiquant une rotation dans le sens des aiguilles d'une montre.
- L'ordre de rendu des tuiles sur des calques de tuiles peut être configuré de plusieurs façons grâce à une nouvelle propriété `renderorder` de l'élément `map`. Les valeurs valides sont `right-down` (par défaut), `right-up`, `left-down` et `left-up`. Dans tous les cas, la carte est rendue ligne par ligne. Seulement supporté pour les cartes orthogonales pour le moment.
- L'ordre de rendu des objets sur des calques d'objets peut être configuré pour soit être trié en utilisant leur coordonnée `y` (comportement précédent et par défaut), soit par l'ordre d'apparence dans le fichier de carte. Ce dernier permet de contrôler l'ordre de rendu manuellement grâce à des actions qui « Montent » et « Descendent » les objets sélectionnés. Ceci est contrôlé par la propriété `draworder` de l'élément `objectgroup`, qui peut soit être `topdown` (défaut) ou `index`.
- Les tuiles peuvent avoir un élément fils `objectgroup` qui peut contenir des objets qui définissent la forme de la collision à utiliser pour cette tuile. Cette information peut être éditée dans le nouvel Éditeur de Collision de Tuiles.

- Les tuiles peuvent avoir une simple animation bouclée qui leur est associée grâce à l’élément fils `animation`. Chaque trame de l’animation réfère un ID de tuile local de ce jeu de tuiles et définit la longueur de la trame en millisecondes. Exemple :

```
<tileset ...>
...
<tile id="[n]">
  <animation>
    <frame tileid="0" duration="100"/>
    <frame tileid="1" duration="100"/>
    <frame tileid="2" duration="100"/>
  </animation>
</tile>
</tileset>
```

20.18 Tiled 0.9

- Le drapeau de visibilité de chaque objet est sauvegardé (1 par défaut) :

```
<object visible="0|1">
```

- Ajout d’informations sur le terrain dans la description des jeux de tuiles (ceci est généralement peu utile pour les jeux) :

```
<tileset ...>
...
<terraintypes>
  <terrain name="Name" tile="local_id"/>
</terraintypes>
<tile id="local_id" terrain="[n],[n],[n],[n]" probability="percentage"/>
...
</tileset>
```

- Ajout d’un support préliminaire pour une projection « échelonnée » (isométrique) (nouvelle valeur pour l’attribut `orientation` de l’élément `map`).
- Un type basique de calque d’images a été ajouté :

```
<imagelayer ...>
<image source="..."/>
</imagelayer>
```

- Ajout d’une forme d’objet elliptique. Elle utilise les mêmes paramètres que les objets rectangulaires, mais est notée en tant qu’ellipse ayant un élément fils :

```
<object ...>
<ellipse/>
</object>
```

- Ajout d’une propriété de carte qui spécifie la couleur d’arrière-plan :

```
<map ... backgroundcolor="#RRGGBB">
```

- Ajout d’un support initial (pas d’interface graphique) pour des images de tuiles individuelles et/ou intégrées (le support est peu important pour le moment car il n’y a aucun moyen d’implémenter ceci dans Tiled Qt, mais c’est possible dans Tiled Java ou avec `pytmxlib`) :

```
<tilesheet ...>
<tile id="[n]">
    <!-- an embedded image -->
    <image format="png">
        <data encoding="base64">
            ...
        </data>
    </image>
</tile>
<tile id="[n]">
    <!-- an individually referenced image for a single tile -->
    <image source="file.png"/>
</tile>
...
</tilesheet>
```

20.19 Tiled 0.8

- Les jeux de tuiles peuvent maintenant avoir des propriétés personnalisées (en utilisant l'élément fils **properties** comme tout le reste).
- Les jeux de tuiles supportent maintenant un décalage de dessin en pixels qui est utilisé lorsque n'importe quelle tuile de ce jeu de tuiles est dessinée. Exemple :

```
<tilesheet name="perspective_walls" tilewidth="64" tileheight="64">
<tileoffset x="-32" y="0"/>
...
</tilesheet>
```

- Ajout du support pour une rotation de tuile en incrément de 90 degrés en utilisant le troisième bit le plus important de l'id de tuile global. Ce nouveau bit indique une « inversion non diagonale », ce qui échange les axes x et y lors du rendu de la tuile.

CHAPITRE 21

Format de Carte JSON

Tiled peut exporter des cartes sous forme de fichiers JSON. Pour ce faire, sélectionnez simplement « File > Export As » et sélectionnez le type de fichier JSON. Vous pouvez exporter en json à partir de la ligne de commande avec l'option **--export-map**.

Les attributs trouvés en format JSON diffère légèrement de ceux trouvés dans *le format de carte tmx*, mais leurs sens restent les mêmes.

Les attributs suivants peuvent être retrouvés dans un fichier Tiled JSON :

21.1 Carte

Attribut	Type	Description
backgroundcolor	string	Couleur en hexadécimal (#RRVVBB or #AARRVVBB) (optionnel)
compressionlevel	int	Le niveau de compression à utiliser pour les données du calque de tuiles (-1 par défaut, ce qui veut dire que l'algorithme par défaut est utilisé)
height	int	Nombre de lignes de tuiles
hexsidelength	int	Longueur d'un côté d'une tuile hexagonale en pixels (seulement pour les cartes hexagonales)
infinite	bool	Si la carte a des dimensions infinies
layers	array	Tableau de Calques
nextlayerid	int	Incrémente automatiquement pour chaque calque
nextobjectid	int	Incrémente automatiquement pour chaque objet placé
orientation	string	<code>orthogonal</code> (orthogonale), <code>isometric</code> (isométrique), <code>staggered</code> (échelonnée) ou <code>hexagonal</code> (hexagonale)
properties	array	Tableau de Propriétés
renderorder	string	<code>right-down</code> (bas-droite, par défaut), <code>right-up</code> (haut-droite), <code>left-down</code> (bas-gauche) ou <code>left-up</code> (haut gauche) (seulement supporté pour les cartes orthogonales pour le moment)
staggeraxis	string	x ou y (seulement pour les cartes échelonnées / hexagonales)
staggerindex	string	<code>odd</code> (impair) ou <code>even</code> (pair) (seulement pour les cartes échelonnées / hexagonales)
tiledversion	string	La version de Tiled utilisée pour sauvegarder le fichier
tileheight	int	Hauteur de la grille de la carte
tilesets	array	Tableau de Jeux de Tuiles
tilewidth	int	Longueur de la grille de la carte
type	string	<code>map</code> (depuis la version 1.0)
version	string	La version du format JSON (auparavant un nombre, sauvegardé en tant que chaîne de caractères depuis la version 1.6)
width	int	Nombre de colonnes de tuiles

21.1.1 Exemple de Carte

```
{
  "backgroundcolor": "#656667",
  "height": 4,
  "layers": [ ],
  "nextobjectid": 1,
  "orientation": "orthogonal",
  "properties": [
    {
      "name": "mapProperty1",
      "type": "string",
      "value": "one"
    },
    {
      "name": "mapProperty2",
      "type": "string",
      "value": "two"
    }
  ]
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }],
    "renderorder": "right-down",
    "tileheight": 32,
    "tilesets": [ ],
    "tilewidth": 32,
    "version": 1,
    "tiledversion": "1.0.3",
    "width": 4
}
```

21.2 Calque

Attribut	Type	Description
chunks	array	Tableau de <i>partitions</i> (optionnel). Seulement pour les calques de tuiles.
compression	string	zlib, gzip, zstd (depuis Tiled 1.3) ou vide (par défaut). Seulement pour les calques de tuiles.
data	tableau chaîne caractères	ou de Tableau d'entiers non-signés (GIDs) ou données encodées en base64. Seulement pour les calques de tuiles.
draworder	string	topdown (haut en bas, par défaut) ou index. Seulement pour les groupes d'objets.
encoding	string	csv (par défaut) ou base64. Seulement pour les calques de tuiles.
height	int	Nombre de lignes. Pareil que la hauteur de la carte pour des cartes à taille fixe.
id	int	ID incrémental - unique pour tous les calques
image	string	Image utilisée pour ce calque. Seulement pour les calques d'images.
layers	array	Tableau de <i>calques</i> . Seulement pour les groupes.
name	string	Nom assigné à ce calque
objects	array	Tableau <i>d'objets</i> . Seulement pour les groupes d'objets.
offsetx	double	Décalage horizontal du calque en pixels (0 par défaut)
offsety	double	Décalage vertical du calque en pixels (0 par défaut)
opacity	double	Valeur comprise entre 0 et 1
parallaxx	double	<i>Facteur de parallaxe</i> horizontal pour ce calque (1 par défaut). (Depuis Tiled 1.5)
parallaxy	double	<i>Facteur de parallaxe</i> vertical pour ce calque (1 par défaut). (Depuis Tiled 1.5)
properties	array	Tableau de <i>Propriétés</i>
startx	int	Coordonnée X avec laquelle le contenu commence (pour les cartes infinies)
starty	int	Coordonnée Y avec laquelle le contenu commence (pour les cartes infinies)
tintcolor	string	<i>Couleur de teinte</i> formatée en hexadécimal (#RRVVBB ou #AARRVVBB) qui est multipliée par tout graphisme dessiné par ce calque ou tout calque enfant (optionnel).
transparentcolor	string	Couleur en hexadécimal (#RRVVBB) (optionnel). Seulement pour les calques d'images.
type	string	tilelayer, objectgroup, imagelayer ou group
visible	bool	Si le calque est visible ou non dans l'éditeur
width	int	Nombre de colonnes. Pareil que la longueur de la carte pour des cartes à taille fixe.
x	int	Décalage horizontal d'un calque en tuiles. Toujours 0.
y	int	Décalage vertical du calque en tuiles. Toujours 0.

21.2.1 Exemple de Calque de Tuiles

The data of a tile layer can be stored as a native JSON array or as base64-encoded and optionally compressed binary data, the same as done in the [TMX format](#). The tiles are referenced using [Global Tile IDs](#).

```
{
  "data": [1, 2, 1, 2, 3, 1, 3, 1, 2, 2, 3, 3, 4, 4, 4, 1],
  "height": 4,
  "name": "ground",
  "opacity": 1,
  "properties": [
    {
      "name": "tileLayerProp",
      "type": "int",
      "value": 1
    }
  ],
  "type": "tilelayer",
  "visible": true,
  "width": 4,
  "x": 0,
  "y": 0
}
```

21.2.2 Exemple de Calque d'Objets

```
{
  "draworder": "topdown",
  "height": 0,
  "name": "people",
  "objects": [],
  "opacity": 1,
  "properties": [
    {
      "name": "layerProp1",
      "type": "string",
      "value": "someStringValue"
    }
  ],
  "type": "objectgroup",
  "visible": true,
  "width": 0,
  "x": 0,
  "y": 0
}
```

21.3 Fragments

Les fragments sont utilisés afin de stocker les données d'un calque de tuiles pour des *cartes infinies*.

Attribut	Type	Description
data	tableau ou chaîne de caractères	Tableau d'entiers non-signés (GIDs) ou données encodées en base64
height	int	Hauteur en tuiles
width	int	Longueur en tuiles
x	int	Coordonnée X en tuiles
y	int	Coordonnée Y en tuiles

21.3.1 Exemple de Fragment

```
{
  "data": [1, 2, 1, 2, 3, 1, 3, 1, 2, 2, 3, 3, 4, 4, 4, 1, ...],
  "height": 16,
  "width": 16,
  "x": 0,
  "y": -16,
}
```

21.4 Objet

Attribut	Type	Description
ellipse	bool	Utilisé pour référencer un objet en tant qu'une ellipse
gid	int	ID de tuile global, seulement si l'objet représente une tuile
height	double	Hauteur en pixels.
id	int	ID incrémental, unique pour tous les objets
name	string	Chaîne de caractères assignée au champ de nom dans l'éditeur
point	bool	Utilisé pour référencer un objet en tant qu'un point
polygon	array	Tableau de <i>Points</i> , si l'objet est un polygone
polyline	array	Tableau de <i>Points</i> , si l'objet est une polyligne
properties	array	Tableau de <i>Propriétés</i>
rotation	double	Angle en degrés dans le sens des aiguilles d'une montre
template	string	Référence à un fichier de modèle, si l'objet est une <i>instance d'un modèle</i>
text	objet JSON	Seulement utilisés pour les objets texte
type	string	Chaîne de caractères assigné à un type de champ dans l'éditeur
visible	bool	Si l'objet est affiché dans l'éditeur.
width	double	Longueur en pixels.
x	double	Coordonnée X en pixels
y	double	Coordonnée Y en pixels

21.4.1 Exemple d'Objet

```
{
  "gid":5,
  "height":0,
  "id":1,
  "name":"villager",
  "properties":[
    {
      "name":"hp",
      "type":"int",
      "value":12
    }],
  "rotation":0,
  "type":"npc",
  "visible":true,
  "width":0,
  "x":32,
  "y":32
}
```

21.4.2 Exemple d'Ellipse

```
{
  "ellipse":true,
  "height":152,
  "id":13,
  "name":"",
  "rotation":0,
  "type":"",
  "visible":true,
  "width":248,
  "x":560,
  "y":808
}
```

21.4.3 Exemple de Rectangle

```
{
  "height":184,
  "id":14,
  "name":"",
  "rotation":0,
  "type":"",
  "visible":true,
  "width":368,
  "x":576,
  "y":584
}
```

21.4.4 Exemple de Point

```
{  
    "point":true,  
    "height":0,  
    "id":20,  
    "name":"",
    "rotation":0,  
    "type":"",
    "visible":true,  
    "width":0,  
    "x":220,  
    "y":350  
}
```

21.4.5 Exemple de Polygone

```
{  
    "height":0,  
    "id":15,  
    "name":"",
    "polygon": [  
        {  
            "x":0,  
            "y":0  
        },  
        {  
            "x":152,  
            "y":88  
        },  
        {  
            "x":136,  
            "y":-128  
        },  
        {  
            "x":80,  
            "y":-280  
        },  
        {  
            "x":16,  
            "y":-288  
        }],  
    "rotation":0,  
    "type":"",
    "visible":true,  
    "width":0,  
    "x":-176,  
    "y":432  
}
```

21.4.6 Exemple de Polyligne

```
{
  "height":0,
  "id":16,
  "name":"",
  "polyline":[
    {
      "x":0,
      "y":0
    },
    {
      "x":248,
      "y":-32
    },
    {
      "x":376,
      "y":72
    },
    {
      "x":544,
      "y":288
    },
    {
      "x":656,
      "y":120
    },
    {
      "x":512,
      "y":0
    }
  ],
  "rotation":0,
  "type":"",
  "visible":true,
  "width":0,
  "x":240,
  "y":88
}
```

21.4.7 Exemple de Texte

```
{
  "height":19,
  "id":15,
  "name":"",
  "text":
  {
    "text":"Hello World",
    "wrap":true
  },
  "rotation":0,
```

(suite sur la page suivante)

(suite de la page précédente)

```

"type": "",  

"visible":true,  

"width":248,  

"x":48,  

"y":136  

}

```

21.5 Texte

Attribut	Type	Description
bold	bool	Si la police doit être en gras (<code>false</code> par défaut)
color	string	Couleur en hexadécimal (#RRVVBB ou #AARRVVBB) (#000000 par défaut)
fontfamily	string	La famille de police utilisée (<code>sans-serif</code> par défaut)
halign	string	Alignement horizontal (<code>center</code> (centré), <code>right</code> (droite), <code>justify</code> (justifié) ou <code>left</code> (gauche, par défaut))
italic	bool	Si la police doit être en italique (<code>false</code> par défaut)
kerning	bool	S'il faut utiliser du crénage lors du placement des caractères (<code>true</code> par défaut)
pixelsize	int	Taille en pixels de la police (16 par défaut)
strikeout	bool	Si le texte doit être barré (<code>false</code> par défaut)
text	string	Texte
underline	bool	Si le texte doit être souligné (<code>false</code> par défaut)
valign	string	Alignement vertical (<code>center</code> (centré), <code>bottom</code> (bas) ou <code>top</code> (haut, par défaut))
wrap	bool	Si le texte est enroulé à l'intérieur des bords de l'objet (<code>false</code> par défaut)

21.6 Jeu de Tuiles

Attribut	Type	Description
backgroundcolor	string	Couleur en hexadécimal (#RRVVBB or #AARRVVBB) (optionnel)
columns	int	Le nombre de colonnes de tuiles du jeu de tuiles
firstgid	int	Le GID de la première tuile du jeu de tuiles
grid	<i>grille de jeu de tuiles JSON</i>	(optionnel)
image	string	Image utilisée pour les tuiles de cette collection
imageheight	int	Hauteur de l'image source en pixels
imagewidth	int	Longueur de l'image source en pixels
margin	int	Marge entre les bords de l'image et la première tuile (en pixels)
name	string	Nom donné à ce jeu de tuiles
objectalignment	string	Alignement à utiliser pour les objets tuiles (<code>unspecified</code> (non spécifié, par défaut), <code>topleft</code> (en haut à gauche), <code>top</code> (en haut), <code>topright</code> (en haut à droite), <code>left</code> (à gauche), <code>center</code> (au centre), <code>right</code> (à droite), <code>bottomleft</code> (en bas à gauche), <code>bottom</code> (en bas) ou <code>bottomright</code> (en bas à droite)) (depuis la 1.4)
properties	array	Tableau de Propriétés
source	string	Le fichier externe qui contient les données de ce jeu de tuiles
spacing	int	Espacement entre deux tuiles adjacentes dans l'image (en pixels)
terrains	array	Tableau de Terrains (optionnel)
tilecount	int	Le nombre de tuiles dans ce jeu de tuiles
tiledversion	string	La version de Tiled utilisée pour sauvegarder le fichier
tileheight	int	La hauteur maximale des tuiles de cette collection
tileoffset	<i>décalage de tuiles de jeu de tuiles JSON</i>	(optionnel)
tiles	array	Tableau de Tuiles (optionnel)
tilewidth	int	Longueur maximale des tuiles de cette collection
transformations	<i>transformations de jeu de tuiles JSON</i>	Transformations autorisées (optionnel)
transparentcolor	string	Couleur en hexadécimal (#RRVVBB) (optionnel)
type	string	<code>tilesheet</code> (pour des fichiers de jeu de tuiles, depuis la 1.0)
version	string	La version du format JSON (auparavant un nombre, sauvegardé en tant que chaîne de caractères depuis la version 1.6)
wangsets	array	Tableau de collections Wang (depuis la 1.1.5)

Tous les jeux de tuiles ont une propriété `firstgid` (premier ID global) qui vous donne l'ID global de sa première tuile (celle qui a pour ID de tuile local 0). Cela vous permet de correspondre les IDs globaux à son vrai jeu de tuiles, et ensuite de calculer l'ID de tuile local en soustrayant `firstgid` de son ID de tuile global. Le premier jeu de tuiles a toujours une valeur `firstgid` égale à 1.

21.6.1 Grille

Spécifie les paramètres communs de la grille utilisés pour les tuiles d'un jeu de tuiles. Visitez [grille](#) dans le Format de Carte TMX.

Attribut	Type	Description
height	int	Hauteur d'une cellule de la grille de tuiles
orientation	string	orthogonal (orthogonale, par défaut) ou isometric (isométrique)
width	int	Longueur d'une cellule de la grille de tuiles

21.6.2 Décalage de Tuile

Visitez [décalage de tuile](#) dans le Format de Carte TMX.

Attribut	Type	Description
x	int	Décalage horizontal en pixels
y	int	Décalage vertical en pixels (positif en bas)

21.6.3 Transformations

Visitez [transformations de jeu de tuiles](#) dans le Format de Carte TMX.

Attribut	Type	Description
hflip	bool	Si les tuiles peuvent être inversées horizontalement
vflip	bool	Si les tuiles peuvent être inversées verticalement
rotate	bool	Si les tuiles peuvent être pivotées par incrément de 90 degrés
preferuntransformed	bool	Si les tuiles non transformées sont préférées, sinon les tuiles transformées sont utilisées pour produire plus de variations

21.6.4 Exemple de Jeu de Tuiles

```
{
  "columns":19,
  "firstgid":1,
  "image": "..\image\fishbaddie_parts.png",
  "imageheight":480,
  "imagewidth":640,
  "margin":3,
  "name":"",
  "properties":[
    {
      "name":"myProperty1",
      "type":"string",
      "value":"myProperty1_value"
    },
    "spacing":1,
    "tilecount":266,
    "tileheight":32,
```

(suite sur la page suivante)

(suite de la page précédente)

```
"tilewidth":32
}
```

21.6.5 Tuile (Définition)

Attribut	Type	Description
animation	array	Tableau de <i>Trames</i>
id	int	ID local de la tuile
image	string	Image représentant cette tuile (optionnel)
imageheight	int	Hauteur de l'image de la tuile en pixels
imagewidth	int	Longueur de l'image de la tuile en pixels
objectgroup	<i>calque</i>	Calque de type groupe d'objets, si des formes de collisions sont spécifiées (optionnel)
probability	double	Pourcentage indiquant la probabilité que cette tuile soit choisie quand elle est en compétition avec d'autres tuiles dans l'éditeur (optionnel)
properties	array	Tableau de <i>Propriétés</i>
terrain	array	Index du terrain pour chaque coin de la tuile (optionnel)
type	string	Le type de la tuile (optionnel)

Un jeu de tuiles qui associe des informations avec chaque tuile, tel que le chemin de son image ou le type de terrain, peut inclure un propriété JSON `tiles`. Chaque tuile a une propriété `id` qui spécifie son ID local dans le jeu de tuiles.

Pour des informations sur le terrain, chaque valeur est un tableau de 4 valeurs dans lequel chaque élément est l'index d'un `terrain` pour un coin de cette tuile. L'ordre est indices est : en haut à gauche, en haut à droite, en bas à gauche, en bas à droite.

Exemple :

```
"tiles": [
  {
    "id":0,
    "properties":[
      {
        "name":"myProperty1",
        "type":"string",
        "value":"myProperty1_value"
      },
      "terrain":[0, 0, 0, 0]
    },
    {
      "id":11,
      "properties":[
        {
          "name":"myProperty2",
          "type":"string",
          "value":"myProperty2_value"
        },
        "terrain":[0, 1, 0, 1]
      },
      {
        "id":12,
        "properties":[
          {
            "name":"myProperty3",
            "type":"string",
            "value":"myProperty3_value"
          },
          "terrain":[1, 0, 1, 0]
        }
      }
    ]
  }
]
```

(suite sur la page suivante)

(suite de la page précédente)

```

"id":12,
"properties":[
{
  "name":"myProperty3",
  "type":"string",
  "value":"myProperty3_value"
},
"terrain":[1, 1, 1, 1]
}
]

```

21.6.6 Trame

Attribut	Type	Description
duration	int	Longueur de la trame en millisecondes
tileid	int	ID local de la tuile représentant cette trame

21.6.7 Terrain

Attribut	Type	Description
name	string	Nom du terrain
properties	array	Tableau de <i>Propriétés</i>
tile	int	ID local de la tuile représentant le terrain

Exemple :

```

"terrains": [
{
  "name":"ground",
  "tile":0
},
{
  "name":"chasm",
  "tile":12
},
{
  "name":"cliff",
  "tile":36
}]

```

21.6.8 Collection Wang

Attribut	Type	Description
colors	array	Tableau de <i>couleurs Wang</i> (depuis la 1.5)
name	string	Nom de la collection Wang
properties	array	Tableau de <i>Propriétés</i>
tile	int	ID local de la tuile représentant la collection Wang
type	string	<i>corner</i> (coin), <i>edge</i> (bord) ou <i>mixed</i> (mixte) (depuis la 1.5)
wangtiles	array	Tableau de <i>tuiles Wang</i>

Couleur Wang

Attribut	Type	Description
color	string	Couleur en hexadécimal (#RRVVBB or #AARRVVBB)
name	string	Nom de la couleur Wang
probability	double	Probabilité utilisée lors de la sélection aléatoire
properties	array	Tableau de <i>Propriétés</i> (depuis la 1.5)
tile	int	ID local de la tuile représentant la couleur Wang

Exemple :

```
{
  "color": "#d31313",
  "name": "Rails",
  "probability": 1,
  "tile": 18
}
```

Tuile Wang

Attribut	Type	Description
tileid	int	ID local de la tuile
wangid	array	Tableau d'index de couleurs Wang (uchar[8])

Exemple :

```
{
  "tileid": 0,
  "wangid": [2, 0, 1, 0, 1, 0, 2, 0]
}
```

21.7 Modèle d'Objet

Un modèle d'objet est écrit dans son propre fichier et est référencé par toute instance de la modèle.

Attribut	Type	Description
type	string	<code>template</code>
tileset	<i>jeu de tuiles</i>	Jeu de tuiles externe utilisé par le modèle (optionnel)
object	<i>objet JSON</i>	L'objet instancié par ce modèle

21.8 Propriété

Attribut	Type	Description
name	string	Nom de cette propriété
type	string	Type de cette propriété (<code>string</code> (chaîne de caractères, par défaut), <code>int</code> (entier), <code>float</code> (flottant), <code>bool</code> (booléen), <code>color</code> (couleur) ou <code>file</code> (fichier, depuis la 0.16, avec les options <code>color</code> et <code>file</code> ajoutées dans la 0.17))
value	value	Valeur de la propriété

21.9 Point

Un point d'un polygone ou d'une polyligne, relatif à la position de l'objet.

Attribut	Type	Description
x	double	Coordonnée X en pixels
y	double	Coordonnée Y en pixels

21.10 Notes de Version

21.10.1 Tiled 1.7

- Les objets *tuile* dans un jeu de tuiles ne sont plus sauvegardés avec un ID incrémental. Ils sont maintenant sauvegardé dans l'ordre d'affichage, ce qui peut être configuré dans Tiled.

21.10.2 Tiled 1.6

- La propriété `version` est maintenant écrite en tant que chaîne de caractères (« 1.6 ») plutôt qu'un nombre (1.5).

21.10.3 Tiled 1.5

- Les propriétés `cornercolors` et `edgecolors` d'une [collection Wang](#) ont été unifiées dans la nouvelle propriété `colors` et un champ `type` a été ajouté.
- Les [couleurs Wang](#) peuvent maintenant stocker des propriétés dans `properties`.
- Ajout de la propriété `transformations` aux [jeux de tuiles](#) (voir [transformations de jeu de tuiles](#)).
- Suppression des propriétés `dflip`, `hflip` et `vflip` des [tuiles Wang](#) (support abandonné).

21.10.4 Tiled 1.4

- Ajout de `objectalignment` à l'objet [jeu de tuiles](#).
- Ajout de `tintcolor` à l'objet [calque](#).

21.10.5 Tiled 1.3

- Ajout de la propriété `editorsettings` aux objets [carte](#) et [jeu de tuiles](#) à la racine, qui est utilisée pour stocker des paramètres spécifiques à l'éditeur qui ne sont généralement pas utiles lors du chargement d'une carte ou d'un jeu de tuiles.
- Ajout du support pour la compression Zstandard pour les données d'un calque de tuiles ("compression" : "zstd" pour les [objets calques de tuiles](#)).
- Ajout de la propriété `compressionlevel` à l'objet [carte](#), qui stocke le niveau de compression à utiliser pour les données des calques de tuiles.

21.10.6 Tiled 1.2

- Ajout de `nextlayerid` à l'objet [carte](#).
- Ajout de `id` à l'objet [calque](#).
- Les tuiles dans un [jeu de tuiles](#) sont maintenant stockées dans un tableau plutôt qu'un objet. Avant les IDs des tuiles étaient stockées en tant que clés chaînes de caractères des objets « tuiles », maintenant elles sont stockées en tant que propriété `id` de chaque objet [Tuile](#).
- Les propriétés personnalisées des tuiles sont maintenant stockées dans chaque [Tuile](#) plutôt qu'être inclus en tant que `tileproperties` dans l'objet [jeu de tuiles](#).
- Les propriétés personnalisées sont maintenant stockées dans un tableau plutôt que dans un objet où le nom des propriétés étaient leur clé. Chaque propriété est maintenant un objet qui stocke son nom, son type et la valeur de la propriété. Les propriétés séparés `propertytypes` et `tilepropertytypes` ont été enlevées.

21.10.7 Tiled 1.1

- Ajout d'un [format de données fragmenté](#), couramment utilisé pour les [cartes infinies](#).
- Les [modèles](#) ont été ajoutés. Les modèles peuvent être stocké en tant que fichiers JSON avec un objet `modèle`.
- Les [jeux de tuiles](#) peuvent maintenant contenir des [Collections de Terrains](#). Elles sont sauvegardées dans le nouvel objet [collection Wang](#) (depuis Tiled 1.1.5).

CHAPITRE 22

Global Tile IDs

Several of the map formats supported by Tiled, including its native [TMX](#) and [JSON](#) map formats, use the same data representation for individual tiles in layers : global tile IDs with flip flags. These GIDs are « global » because they may refer to a tile from any of the tilesets used by the map, rather than being local to a specific tileset. To get at a specific tile from a GID, you will first need to extract and clear the flip flags, then you will need to determine which tileset the tile belongs to, and which tile within the tileset it is.

Note : Despite the « global » name, GIDs are only global within a single map. A given tile may have a different GID in a different map, if that map has different tilesets, or has its tilesets in a different order.

22.1 Tile Flipping

The highest four bits of the 32-bit GID are flip flags, and you will need to read and clear them before you can access the GID itself to identify the tile.

Bit 32 is used for storing whether the tile is horizontally flipped, bit 31 is used for the vertically flipped tiles. In orthogonal and isometric maps, bit 30 indicates whether the tile is flipped (anti) diagonally, which enables tile rotation, and bit 29 can be ignored. In hexagonal maps, bit 30 indicates whether the tile is rotated 60 degrees clockwise, and bit 29 indicates 120 degrees clockwise rotation.

Note : Even if you're parsing a non-hexagonal map, remember to clear bit 29 after you've read the flags. Tiled keeps and outputs that flag even if the map orientation is changed. If not cleared, you may get an invalid tile ID.

When rendering an orthographic or isometric tile, the order of operations matters. The diagonal flip is done first, followed by the horizontal and vertical flips. The diagonal flip should flip the bottom left and top right corners of the tile, and can be thought of as an x/y axis swap. For hexagonal tiles, the order does not matter.

22.2 Mapping a GID to a Local Tile ID

Every tileset has its own, independent local tile IDs, typically (but not always) starting at 0. To avoid conflicts within maps using multiple tilesets, GIDs are assigned in sequence based on the size of each tileset. Each tileset is assigned a `firstgid` within the map, this is the GID that the tile with local ID 0 in the tileset would have.

To figure out which tileset a tile belongs to, find the tileset that has the largest `firstgid` that is smaller than or equal to the tile's GID. Once you have identified the tileset, subtract its `firstgid` from the tile's GID to get the local ID of the tile within the tileset.

Note : The `firstgid` of the first tileset is always 1. A GID of 0 in a layer means that cell is empty.

As an example, here's an excerpt from a TMX file with three tilesets :

```
<tileset firstgid="1" source="TilesetA.tsx"/>
<tileset firstgid="65" source="TilesetB.tsx"/>
<tileset firstgid="115" source="TilesetC.tsx"/>
```

In this map, tiles with GIDs 1-64 would be part of TilesetA, tiles with GIDs 65-114 would be part of TilesetB, and tiles with GIDs 115 and above would be part of tileset C. A tile with GID 72 would be part of TilesetB since TilesetB has the largest `firstgid` that's less than or equal to 72, and its local ID would be 7 (72-65).

22.3 Code example

The following C++ pseudo-code, using TMX as an example, should make it all clear, it deals with flags and deduces the appropriate tileset :

```
// Bits on the far end of the 32-bit global tile ID are used for tile flags
const unsigned FLIPPED_HORIZONTALLY_FLAG = 0x80000000;
const unsigned FLIPPED_VERTICALLY_FLAG = 0x40000000;
const unsigned FLIPPED_DIAGONALLY_FLAG = 0x20000000;
const unsigned ROTATED_HEXAGONAL_120_FLAG = 0x10000000;

...

// Extract the contents of the <data> element
string tile_data = ...

// If the data is encoded and compressed, decode and decompress:
unsigned char *data = decompress(base64_decode(tile_data));

unsigned tile_index = 0;

// Here you should check that the data has the right size
// (map_width * map_height * 4)

for (int y = 0; y < map_height; ++y) {
    for (int x = 0; x < map_width; ++x) {
        //Read the GID in little-endian byte order:
        unsigned global_tile_id = data[tile_index] |
            data[tile_index + 1] << 8 |
```

(suite sur la page suivante)

(suite de la page précédente)

```
        data[tile_index + 2] << 16 |
        data[tile_index + 3] << 24;
    tile_index += 4;

    // Read out the flags
    bool flipped_horizontally = (global_tile_id & FLIPPED_HORIZONTALLY_FLAG);
    bool flipped_vertically = (global_tile_id & FLIPPED_VERTICALLY_FLAG);
    bool flipped_diagonally = (global_tile_id & FLIPPED_DIAGONALLY_FLAG);
    bool rotated_hex120 = (global_tile_id & ROTATED_HEXAGONAL_120_FLAG);

    // Clear all four flags
    global_tile_id &= ~(FLIPPED_HORIZONTALLY_FLAG |
                        FLIPPED_VERTICALLY_FLAG |
                        FLIPPED_DIAGONALLY_FLAG |
                        ROTATED_HEXAGONAL_120_FLAG);

    // Resolve the tile
    for (int i = tileset_count - 1; i >= 0; --i) {
        Tileset *tileset = tilesets[i];

        if (tileset->first_gid() <= global_tile_id) {
            tiles[y][x] = tileset->tileAt(global_tile_id - tileset->first_gid());
            break;
        }
    }
}
```

(Since the above code was put together on this wiki page and can't be directly tested, please make sure to report any errors you encounter when basing your parsing code on it, thanks!)

CHAPITRE 23

Création de Script

23.1 Introduction

Tiled peut être étendu avec l'utilisation de JavaScript. Visitez [l'API de Création de Script Tiled](#) pour une référence de toutes les fonctionnalités disponibles.

Les définitions TypeScript de l'API sont disponibles dans le paquet NPM [@mapeditor/tiled-api](#), qui peut fournir une auto-complétion dans votre éditeur. La référence de l'API est générée suivant ces définitions.

Au lancement, Tiled va exécuter n'importe quel fichier de script présent dans les *dossiers d'extension*. De plus il est possible de lancer des scripts directement depuis *la console*. Tous les scripts partagent un seul contexte JavaScript.

Note : Quelques exemples de scripts et des liens vers des extensions de Tiled sont fournis dans le répertoire d'Extensions Tiled : <https://github.com/mapeditor/tiled-extensions>

Note : Le support pour l'entièreté de l'API de création de script pour les fonctionnalités de ECMAScript 7 est seulement disponible pour les versions de Tiled basées sur Qt 5.12 ou ultérieur. Pour le moment, cela exclut les versions pour Windows XP et snap.

23.1.1 Extensions Scriptées

Les extensions peuvent être placées dans une location pour un système spécifique ou un *projet spécifique*.

Le dossier pour les extensions pour un système spécifique peut être ouvert depuis l'onglet Greffons dans la *fenêtre de Préférences*. La location usuelle de chaque plateforme supportée est comme ceci :

Windows	C:/Users/<USER>/AppData/Local/Tiled/extensions/
macOS	~/Library/Preferences/Tiled/extensions/
Linux	~/.config/tiled/extensions/

Les dossier spécifiques à chaque projet est nommé « extensions » par défaut, relativement au dossier du fichier `tiled-project`, mais cela peut être changé dans les *Propriétés du Projet*.

Avertissement : Depuis Tiled 1.7, les extensions spécifiques à un projet sont seulement activées par défaut pour les projets que vous avez créés. Lorsque vous ouvrez tout autre projet, un popup vous notifiera lorsqu'un projet a un dossier d'extensions scriptées, qui vous permet ensuite d'activer les extensions pour ce projet.

Faites toujours attention lors de l'activation d'extensions pour des projets que vous n'avez pas créés, car les extensions ont accès à vos fichiers et peuvent lancer des processus.

Une extension peut être placée directement dans un dossier `extensions`, ou dans un sous-dossier. Tous les scripts trouvés dans ces dossiers sont lancés au lancement de l'application.

Quand tout script chargé est changé ou lorsque n'importe quel fichier est ajouté/enlevé du dossier `extensions`, le moteur de script est automatiquement réinitialisé et les scripts sont rechargés. De cette façon, il n'y a pas besoin de relancer Tiled lors de l'installation d'extensions. C'est aussi plus rapide lors de l'édition d'un script jusqu'à ce qu'il marche comme prévu.

Mis à part les scripts, les extensions peuvent inclure des images qui peuvent être utilisées en tant qu'icônes pour les actions scriptées ou les outils.

23.1.2 Vue Console

Dans la vue Console (*Vue > Vues et Barres d'Outils > Console*) vous allez voir une entrée de texte où vous pouvez écrire ou coller des scripts pour les évaluer.

Vous pouvez utiliser les touches Haut/Bas pour naviguer entre les expressions de scripts entrées précédemment.

23.2 Référence de l'API

Visitez l'API de Création de Script Tiled.

La variable globale suivant n'est pour le moment pas documentée dans la documentation générée, car elle génère des conflits avec des types de nodejs :

`_filename` Le chemin du fichier couramment évalué. Seulement disponible pendant l'évaluation initiale du fichier et non lorsque des fonctions de ce fichier sont appelées plus tard. Si vous en avez besoin, copiez cette valeur dans une variable locale.