Squirrel Battle

Yohann D'ANELLO, Mathilde DEPRES, Nicolas MARGULIES, Char

déc. 04, 2020

Développer

1 Installation d'un environnement de développement **3**

2 Exécution des tests **5**

3 Gestion de l'affichage **7**
- 3.1 Affichage des menus . . . 7
- 3.2 Affichage de la carte . . . 7
- 3.3 Affichage des statistiques . . . 7
- 3.4 Affichage de l'historique . . . 8

4 Déploiement du projet **9**
- 4.1 PyPI . . . 9
  - 4.1.1 Définition du paquet . . . 9
  - 4.1.2 Installation locale du paquet . . . 10
  - 4.1.3 Génération des binaires . . . 10
  - 4.1.4 Publier sur PyPI . . . 10
- 4.2 Publier dans l'AUR . . . 11
  - 4.2.1 Fonctionnement du packaging . . . 11
  - 4.2.2 Mettre à jour . . . 13
- 4.3 Construction du paquet Debian . . . 13
- 4.4 Structure du paquet . . . 13
- 4.5 Mettre à jour le paquet . . . 13
- 4.6 Construire le paquet . . . 13

5 Documentation **15**
- 5.1 Générer localement la documentation . . . 15
- 5.2 Documentation externe . . . 15

6 Installation client **17**
- 6.1 Installation . . . 17
  - 6.1.1 Depuis PIP . . . 17
  - 6.1.2 Sur Arch Linux . . . 17
  - 6.1.3 Sur Ubuntu/Debian . . . 18

7 Règles du jeu **19**

8 Carte **21**

8.1 Tuiles . . . 21
8.1.1 Vide . . . 21
8.1.2 Sol . . . 21
8.1.3 Mur . . . 22

9 Entités **23**
9.1 Joueur . . . 23
9.1.1 Déplacement . . . 23
9.1.2 Expérience . . . 24
9.2 Monstres . . . 24
9.2.1 Hérisson . . . 24
9.2.2 Tigre . . . 24
9.2.3 Lapin . . . 24
9.2.4 Nounours . . . 24
9.3 Objets . . . 25
9.3.1 Bombe . . . 25
9.3.2 Cœur . . . 25
9.4 Entité . . . 25
9.5 Entité attaquante . . . 26

10 Pack de textures **27**
10.1 Pack ASCII . . . 27
10.2 Pack Écureuil . . . 28

11 Paramètres **29**

12 Résolution d'erreurs **31**
12.1 Émojis . . . 31
12.1.1 Sous Arch Linux . . . 31
12.1.2 Sous Ubuntu/Debian . . . 31

CHAPITRE 1

Installation d'un environnement de développement

Il est toujours préférable de travailler dans un environnement Python isolé du reste de son instalation.

1. **Installation des dépendances de la distribution.** Vous devez déjà installer Python et le module qui permet de créer des environnements virtuels. On donne ci-dessous l'exemple pour une distribution basée sur Debian, mais vous pouvez facilement adapter pour ArchLinux ou autre.

```
$ sudo apt update
$ sudo apt install --no-install-recommends -y python3-setuptools python3-venv python3-
↪dev git
```

2. **Clonage du dépot** là où vous voulez :

```
$ git clone git@gitlab.crans.org:ynerant/squirrel-battle.git && cd squirrel-battle
```

3. **Création d'un environment de travail Python décorrélé du système.** On n'utilise pas *–system-site-packages* ici pour ne pas avoir des clashs de versions de modules avec le système.

```
$ python3 -m venv env
$ source env/bin/activate  # entrer dans l'environnement
(env)$ pip3 install -r requirements.txt
(env)$ deactivate  # sortir de l'environnement
```

Le lancement du jeu se fait en lançant la commande `python3 main.py`.

CHAPITRE 2

Exécution des tests

Note : La documentation va être revue ici.

Les tests sont gérés par `pytest` dans le module `squirrelbattle.tests`.

`tox` est un outil permettant de configurer l'exécution des tests. Ainsi, après installation de tox dans votre environnement virtuel via `pip install tox`, il vous suffit d'exécuter `tox -e py3` pour lancer les tests et `tox -e linters` pour vérifier la syntaxe du code.

CHAPITRE 3

Gestion de l'affichage

L'intégralité de l'affichage du jeu est géré grâce à la bibliothèque native de Python curses.

Avertissement : Plus de documentation à venir.

3.1 Affichage des menus

Pas encore documenté.

3.2 Affichage de la carte

Pas encore documenté.

3.3 Affichage des statistiques

Pas encore documenté.

3.4 Affichage de l'historique

Pas encore documenté.

CHAPITRE 4

Déploiement du projet

À chaque nouvelle version du projet, il est compilé et déployé dans PyPI, dans l'AUR et un paquet Debian est créé, voir la page d'installation.

4.1 PyPI

4.1.1 Définition du paquet

La documentation sur le packaging dans PyPI est disponible ici.

Le fichier setup.py contient l'ensemble des instructions d'installation du paquet ainsi que des détails à fournir à PyPI :

```
#!/usr/bin/env python3
import os

from setuptools import find_packages, setup

with open("README.md", "r") as f:
    long_description = f.read()

setup(
    name="squirrel-battle",
    version="3.14.1",
    author="ÿnérant, eichhornchen, nicomarg, charlse",
    author_email="squirrel-battle@crans.org",
    description="Watch out for squirrel's knives!",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://gitlab.crans.org/ynerant/squirrel-battle",
    packages=find_packages(),
    license='GPLv3',
    classifiers=[
        "Development Status :: 4 - Beta",
```

(suite sur la page suivante)

(suite de la page précédente)

```
        "Environment :: Console :: Curses",
        "License :: OSI Approved :: GNU General Public License v3 (GPLv3)",
        "Natural Language :: French",
        "Operating System :: OS Independent",
        "Programming Language :: Python :: 3",
        "Programming Language :: Python :: 3.6",
        "Programming Language :: Python :: 3.7",
        "Programming Language :: Python :: 3.8",
        "Programming Language :: Python :: 3.9",
        "Topic :: Games/Entertainment",
    ],
    python_requires='>=3.6',
    include_package_data=True,
    package_data={"squirrelbattle": ["assets/*"]},
    entry_points={
        "console_scripts": [
            "squirrel-battle = squirrelbattle.bootstrap:Bootstrap.run_game",
        ]
    }
)
```

Ce fichier contient le nom du paquet, sa version, l'auteur et son contact, sa description en une ligne et sa description longue, le lien d'accueil du projet, sa licence, ses classificateurs et son exécutable.

Le paramètre `entry_points` définit un exécutable nommé `squirrel-battle`, qui permet de lancer le jeu.

4.1.2 Installation locale du paquet

L'installation du paquet localement dans son environnement Python (virtuel ou non) peut se faire en exécutant `pip install -e .`.

4.1.3 Génération des binaires

Les paquets `setuptools` (`python3-setuptools` pour APT, `python-setuptools` pour pacman) et `wheel` (`python3-wheel` pour APT, `python-wheel` pour pacman) sont nécessaires. Une fois installés, il faut appeler la commande :

```
python3 setup.py sdist bdist_wheel
```

Une fois cela fait, le dossier `dist/` contiendra les archives à transmettre à PyPI.

4.1.4 Publier sur PyPI

Il faut avant tout avoir un compte sur PyPI. Dans votre compte PyPI, il faut générer un jeton d'accès API.

Dans le fichier `.pypirc` dans le répertoire principal de l'utilisateur, il faut ajouter le jeton d'accès :

```
[pypi]
  username = __token__
  password = pypi-my-pypi-api-access-token
```

Cela permet de s'authentifier directement par ce jeton.

Ensuite, il faut installer `twine`, qui permet de publier sur PyPI.

Il suffit ensuite d'appeler :

```
twine upload dist/*
```

pour envoyer le paquet sur PyPI.

Note : À des fins de tests, il est possible d'utiliser le dépôt https://test.pypi.org. Les différences sont au niveau de l'authentification, où il faut l'en-tête `[testpypi]` dans le `.pypirc`, et il faut envoyer le paquet avec `twine upload --repository testpypi dist/`.

4.2 Publier dans l'AUR

4.2.1 Fonctionnement du packaging

Deux paquets sont publiés dans l'AUR (Arch User Repository) :

— python-squirrel-battle
— python-squirrel-battle-git

Le packaging dans Arch Linux se fait en commitant un fichier `PKGBUILD` dans le dépôt à l'adresse `ssh://aur@aur.archlinux.org/packagename.git`, en remplaçant `packagename` par le nom du paquet.

Le second paquet compile directement le jeu à partir de la branche `master` du dépôt Git. Le fichier `PKGBUILD` dispose de cette structure :

```
# Maintainer: Yohann D'ANELLO <squirrel-battle@crans.org>

pkgbase=squirrel-battle
pkgname=python-squirrel-battle-git
pkgver=3.14.1
pkgrel=1
pkgdesc="Watch out for squirrel's knives!"
arch=('any')
url="https://gitlab.crans.org/ynerant/squirrel-battle"
license=('GPLv3')
depends=('python')
makedepends=('python-setuptools')
depends=('noto-fonts-emoji')
checkdepends=('python-tox')
ssource=("git+https://gitlab.crans.org/ynerant/squirrel-battle.git")
sha256sums=("SKIP")

pkgver() {
  cd pkgbase
  git describe --long --tags | sed -r 's/^v//;s/([^-]*-g)/r\1/;s/-/./g'
}
build() {
  cd $pkgbase
  python setup.py build
}

check() {
  cd $pkgbase
  tox -e py3
  tox -e linters
```

(suite sur la page suivante)

(suite de la page précédente)

```
}

package() {
  cd $pkgbase
  python setup.py install --skip-build \
                          --optimize=1 \
                          --root="${pkgdir}"
  install -vDm 644 README.md \
    -t "${pkgdir}/usr/share/doc/${pkgname}"
  install -vDm 644 LICENSE -t "${pkgdir}/usr/share/licenses/${pkgname}"
}
```

Ces instructions permettent de cloner le dépôt, l'installer et exécuter des tests, en plus de définir les attributs du paquet.

Le fichier `PKGBUILD` du paquet `python-squirrel-battle`, synchronisé avec les releases, est plus ou moins similaire :

```
# Maintainer: Yohann D'ANELLO <squirrel-battle@crans.org>

pkgbase=squirrel-battle
pkgname=python-squirrel-battle
pkgver=3.14.1
pkgrel=1
pkgdesc="Watch out for squirrel's knives!"
arch=('any')
url="https://gitlab.crans.org/ynerant/squirrel-battle"
license=('GPLv3')
depends=('python')
makedepends=('python-setuptools')
depends=('noto-fonts-emoji')
checkdepends=('python-tox')
source=("https://gitlab.crans.org/ynerant/squirrel-battle/-/archive/v3.14.1/$pkgbase-v
↪$pkgver.tar.gz")
sha256sums=("6090534d598c0b3a8f5acdb553c12908ba8107d62d08e17747d1dbb397bddef0")

build() {
  cd $pkgbase-v$pkgver
  python setup.py build
}

check() {
  cd $pkgbase-v$pkgver
  tox -e py3
  tox -e linters
}

package() {
  cd $pkgbase-v$pkgver
  python setup.py install --skip-build \
                          --optimize=1 \
                          --root="${pkgdir}"
  install -vDm 644 README.md \
    -t "${pkgdir}/usr/share/doc/${pkgname}"
  install -vDm 644 LICENSE -t "${pkgdir}/usr/share/licenses/${pkgname}"
}
```

Il se contente ici de télécharger l'archive de la dernière release, et de travailler dessus.

4.2.2 Mettre à jour

Pour mettre à jour le dépôt, une fois les dépôts `ssh://aur@aur.archlinux.org/python-squirrel-battle.git` et `ssh://aur@aur.archlinux.org/python-squirrel-battle-git.git` clonés, il suffit de mettre à jour le paramètre `pkgver` pour la bonne version, de régénérer le fichier `.SRCINFO` en faisant `makepkg --printsrcinfo > .SRCINFO`, puis de committer/pousser.

4.3 Construction du paquet Debian

4.4 Structure du paquet

L'ensemble des instructions pour construire le paquet Debian est situé dans le dossier `debian/`.

Le fichier `changelog` est à modifier à chaque nouvelle version, le fichier `compat` contient la version minimale de Debian requise (`10` pour Debian Buster), le fichier `copyright` contient la liste des fichiers distribués sous quelle licence (ici GPLv3), le fichier `control` contient les informations du paquet, le fichier `install` les fichiers de configuration à installer (ici le fix de l'affichage de l'écurueil), et enfin le fichier `rules` l'ensemble des instructions à exécuter pour installer.

Le paquet `fonts-noto-color-emoji` est en dépendance pour le bon affichage des émojis.

4.5 Mettre à jour le paquet

Pour changer la version du paquet, il faut ajouter des lignes dans le fichier `changelog`.

4.6 Construire le paquet

Il faut partir d'une installation de Debian.

D'abord on installe les paquets nécessaires :

```
apt update
apt --no-install-recommends install build-essential debmake dh-python debhelper␣
↪python3-all python3-setuptools
```

On peut ensuite construire le paquet :

```
dpkg-buildpackage
mkdir build && cp ../*.deb build/
```

Le paquet sera installé dans `build/python3-squirrel-battle_3.14.1_all.deb`.

Le paquet Debian est construit par l'intégration continue Gitlab et ajouté à chaque release.

CHAPITRE 5

Documentation

La documentation est gérée grâce à Sphinx. Le thème est le thème officiel de ReadTheDocs `sphinx-rtd-theme`.

5.1 Générer localement la documentation

On commence par se rendre au bon endroit et installer les bonnes dépendances :

```
cd docs
pip install -r requirements.txt
```

La documentation se génère à partir d'appels à `make`, selon le type de documentation voulue.

Par exemple, `make html` construit la documentation web, `make latexpdf` construit un livre PDF avec cette documentation.

5.2 Documentation externe

À chaque commit, un webhook est envoyé à readthedocs.io, qui construit tout seul la documentation Sphinx, la publiant à l'adresse squirrel-battle.readthedocs.io.

De plus, les documentations sont sauvegardées à chaque release taguée.

CHAPITRE 6

Installation client

6.1 Installation

Différents paquets sont déployés, dans PyPI pour tout système utilisant Python, un paquet Debian et un paquet Arch Linux.

6.1.1 Depuis PIP

Le projet *Squirrel Battle* est déployé dans PyPI. Il suffit d'installer Squirrel Battle en exécutant :

```
pip install --user squirrel-battle
```

Les mises à jour s'obtiennent également via PIP en exécutant :

```
pip install --user --upgrade squirrel-battle
```

Le jeu peut se lancer ensuite en exécutant la commande `squirrel-battle`.

Toutefois, le paquet PyPI n'inclut pas les polices d'émojis. Il est recommandé d'installer des polices telles que `noto-fonts-emoji` afin de prendre en charge les émojis dans votre terminal.

6.1.2 Sur Arch Linux

Deux paquets sont publiés dans l'AUR (Arch User Repository) :

— python-squirrel-battle
— python-squirrel-battle-git

Le premier paquet est mis à jour à chaque nouvelle version déployée, le second est utile pour des fins de développement et est en permanence à jour avec la branche `master` du Git.

Les deux ne sont pas présents dans les dépôts officiels de Arch Linux, mais vous pouvez les récupérer avec un outil tel que yay.

Les paquets incluent la dépendance `noto-fonts-emoji`, qui permet d'afficher les émojis dans le terminal.

Le jeu peut être ensuite lancé via la commande `squirrel-battle`.

6.1.3 Sur Ubuntu/Debian

Un paquet est généré par l'intégration continue de Gitlab à chaque commit. Ils sont également attachés à chaque nouvelle release.

Il dépend du paquet `fonts-noto-color-emoji`, permettant d'afficher les émojis dans le terminal. Il peut être installé via APT.

Pour installer ce paquet, il suffit de le télécharger et d'appeler `dpkg` :

```
dpkg -i python3-squirrelbattle_3.14.1_all.deb
```

Ce paquet inclut un patch pour afficher les émojis écureuil correctement.

Après cela, le jeu peut être lancé grâce à la commande `squirrel-battle`.

CHAPITRE 7

Règles du jeu

Dans *Squirrel Game*, le joueur incarne un écureuil coincé dans un donjon, prêt à tout pour s'en sortir. Sa vision de rongeur lui permet d'observer l'intégralité de la carte, et à l'aide d'"objets, il va pouvoir affronter les monstres présents dans le donjon et gagner en expérience et en force.

Le jeu fonctionne par niveau. À chaque niveau `n` du joueur, entre `3n` et `7n` entités apparaissent aléatoirement sur la carte.

En tuant des ennemis, ce qu'il parvient à faire en fonçant directement sur eux ayant mangé trop de noisettes (ou étant armé d'un couteau), l'écureuil va pouvoir gagner en expérience et au fur et à mesure qu'il monte de niveau, a force augmentera.

Arriverez-vous à sauver ce malheureux petit écureuil perdu ?

Bon courage sachant que le jeu est sans fin ...

CHAPITRE 8

Carte

Dans Squirrel game, le joueur se déplace dans un donjon, constitué de plusieurs cartes. Pour le moment, le jeu se déroule sur une unique carte pré-définie, non générée aléatoirement.

Une carte est un rectangle composé de *tuiles*.

La carte est chargée depuis sa représentation ASCII dans un fichier texte.

Au lancement du jeu, une quantité aléatoire d'"entités sont générées et placées aléatoirement sur la carte.

8.1 Tuiles

Une tuile représente une case du jeu, avec ses différentes propriétés physiques. On compte actuellement 3 types de tuiles :

8.1.1 Vide

Le vide est représenté par un espace vide quelque que soit le pack de textures utilisé. Cette tuile n'est utilisée que pour délimiter les bords de la carte, aucune entité ne peut se trouver sur cette tuile.

8.1.2 Sol

Le sol représente les emplacements où les entités peuvent se déplacer librement. Il est représenté par un point `.` dans le pack de textures ASCII et par deux caractères rectangulaires blancs dans le pack de textures écureuil.

8.1.3 Mur

Les murs délimitent les salles du donjon. Personne ne peut les traverser. Ils sont représentés par un dièse # dans le pack de textures ASCII et par une brique carrée dans le pack de textures écureuil.

CHAPITRE 9

Entités

9.1 Joueur

Le joueur est une entité attaquante, contrôlée par l'utilisateur humain.

Il est représenté dans le pack de textures ASCII par le caractère `@`, et dans le pack de textures écureuil par le fameux émoji écureuil .

En plus des attributs d'une entité attaquante, le joueur dispose des atrributs supplémentaires :

— `current_xp: int`
Correspond à l'expérience accumulée par le joueur depuis le dernier niveau obtenu.

— `max_xp: int`
Expérience requise au joueur pour changer de niveau. Vaut 10 fois le niveau actuel.

— `inventory: List[Item]`
Contient l'ensemble des objets détenus par le joueur.

9.1.1 Déplacement

Selon les paramètres, il est possible de bouger le joueur dans les 4 directions en appuyant sur `z`, `q`, `s`, `d` ou sur les flèches directionnelles.

Le joueur se retrouvera bloqué s'il avance contre un mur. Si il avance sur un objet, alors il prend l'"objet et avance sur la case.

S'il rencontre une autre entité attaquante, alors il frappe l'entité en infligeant autant de dégâts qu'il n'a de force. À chaque fois qu'une entité est tuée, le joueur gagne aléatoirement entre 3 et 7 points d'expérience.

9.1.2 Expérience

À chaque monstre tué, le joueur gagne entre 3 et 7 points d'expérience aléatoirement. Lorsque le joueur atteint la quantité d'expérience requise pour monter de niveau, le joueur gagne un niveau, regagne toute sa vie, consomme son expérience et la nouvelle quantité d'expérience requise est 10 fois le niveau actuel. De plus, entre 5 et 10 fois le niveau actuel entités apparaissent aléatoirement sur la carte à la montée de niveau. Enfin, le joueur gagne en force en montant de niveau.

9.2 Monstres

Chaque monstre est une entité attaquante, et hérite donc de ses attributs.

À chaque tick de jeu, chaque monstre se déplace d'une case, si possible. Si le monstre est loin du joueur, ce déplacement est fait aléatoirement. Sinon, si le monstre est à moins de 5 cases du joueur, alors il se dirige au plus vite sur le joueur pour le frapper selon l'algorithme de Dijkstra, et s'il est suffisamment proche frappe le joueur et lui fait autant de dégâts qu'il n'a de force.

On dénombre actuellement 4 types de monstres :

9.2.1 Hérisson

Son nom est fixé à *hedghog*. Il a par défaut une force à **3** et **10** points de vie.

Dans le pack de textures ASCII, il est représenté par le caractère `*`.

Dans le pack de textures écureuil, il est représenté par l'émoji .

9.2.2 Tigre

Son nom est fixé à *tiger*. Il a par défaut une force à **2** et **20** points de vie.

Dans le pack de textures ASCII, il est représenté par le caractère `n`.

Dans le pack de textures écureuil, il est représenté par l'émoji .

9.2.3 Lapin

Son nom est fixé à *rabbit*. Il a par défaut une force à **1** et **15** points de vie.

Dans le pack de textures ASCII, il est représenté par le caractère `Y`.

Dans le pack de textures écureuil, il est représenté par l'émoji .

9.2.4 Nounours

Son nom est fixé à *teddy_bear*. Il n'a pas de force et **50** points de vie.

Dans le pack de textures ASCII, il est représenté par le caractère `8`.

Dans le pack de textures écureuil, il est représenté par l'émoji .

9.3 Objets

Un objet est une entité présente sur la carte que le joueur peut ramasser. Il lui suffit pour cela de s'approcher, et une fois sur la case de l'objet, celui-ci est placé dans l'inventaire.

Un objet dispose de deux paramètres :

— `held: bool`
Indique si l'objet est placé dans l'inventaire ou s'il est au sol.

— `held_by: Optional[Player]`
Si l'objet est dans l'inventaire, renvoie son propriétaire.

Deux types d'objets sont pour l'instant présents :

9.3.1 Bombe

Une bombe est un objet que l'on peut ramasser. Une fois ramassée, elle est placée dans l'inventaire. Le joueur peut ensuite lâcher la bombe, qui fera alors 3 dégâts à toutes les entités attaquantes situées à moins de une case.

Elle est représentée dans le pack de textures ASCII par le caractère ○ et dans le pack de textures écureuil par l'émoji .

Note : La gestion de l'inventaire n'ayant pas encore été implémentée, il n'est à l'heure actuelle pas possible de lancer une bombe.

9.3.2 Cœur

Une cœur est un objet que l'on ne peut pas ramasser. Dès que le joueur s'en approche, il est régénéré automatiquement de 3 points de vie, et le cœur disparaît.

Elle est représentée dans le pack de textures ASCII par le caractère et dans le pack de textures écureuil par l'émoji .

9.4 Entité

Une entité est un élément placé sur la carte. Ce peut être le joueur, un monstre ou bien un objet sur la carte. Chaque entité dispose des attributs suivants :

— `name: str`
Il s'agit du type de l'entité.

— `y: int`

— `x: int`
Cela représente les coordonnées de l'entité sur la carte.

— `map: Map`
Il s'agit de la carte sur laquelle est placée l'entité.

Il existe à l'heure actuelle deux types d'entité : une *entité attaquante* ou bien un objet.

9.5 Entité attaquante

Une entité attaquante (`FightingEntity`) est un type d'entités représentant les personnages présents sur la carte, pouvant alors se battre. Ce peut être un monstre ou bien le joueur.

Elles disposent toutes, en plus des paramètres d'entité, des attributs suivants :

— `maxhealth: int`
Représente la vie maximale de l'entité, qui est aussi la vie de départ.
— `health: int`
Représente la vie actuelle de l'entité.
— `strength: int`
Représente la force de l'entité, le nombre de dégâts à faire à chaque coup.
— `intelligence: int`
— `charisma: int`
— `dexterity: int`
— `constitution: int`
Tous ces paramètres sont des statistiques de l'entité, n'ayant pas de réelle influence pour le moment.
— `level: int`
Niveau de l'entité.

Chaque type d'entité disposera de ses propres attributs de départ.

On considère une entité comme morte à partir du moment où sa vie descend en-dessous de 0 point de vie. À ce moment-là, l'entité est retirée de la carte.

Lorsqu'une entité en frappe une autre, celle-ci inflige autant de dégâts qu'elle n'a de force, et autant de points de vie sont perdus.

CHAPITRE 10

Pack de textures

Chaque entité et chaque tuile de la carte est associé à un caractère pour être affiché dans le terminal. Cependant, afin de pouvoir proposer plusieurs expériences graphiques (notamment en fonction du support des émojis), différents packs de textures sont proposés.

Il est possible de changer de pack dans les paramètres.

Les packs de textures peuvent influencer la taille que prennent les tuiles, en raison du fait que les émojis ne sont pas monospace.

Les packs de textures sont au nombre de deux :

10.1 Pack ASCII

Chaque tuile fait un caractère de large.

- **Tuiles**
  - Vide : *espace*
  - Mur : `#`
  - Sol : `.`
- **Entités**
  - Joueur : `@`
  - Hérisson : `*`
  - Cœur :
  - Bombe : `o`
  - Lapin : `Y`
  - Tigre : `n`
  - Nounours : `8`

10.2 Pack Écureuil

Chaque tuile fait 2 caractères de large pour afficher les émojis proprement.

- **Tuiles**
  - Vide : *espace*
  - Mur :
  - Sol :
- **Entités**
  - Joueur :
  - Hérisson :
  - Cœur :
  - Bombe :
  - Lapin :
  - Tigre :
  - Nounours :

CHAPITRE 11

Paramètres

Pas encore documenté.

CHAPITRE 12

Résolution d'erreurs

12.1 Émojis

Le jeu s'exécutant en terminal, il est courant d'obtenir des problèmes d'affichage. Sous Windows, les émojis s'affichent normalement correctement. Il suffit en général d'installer les bons paquets de police.

12.1.1 Sous Arch Linux

Il est recommandé d'utiliser le terminal *xfce4-terminal*. Il suffit d'installer le paquets de polices :

```
sudo pacman -Sy noto-fonts-emoji
```

Le jeu doit ensuite se lancer normalement sans action supplémentaire.

12.1.2 Sous Ubuntu/Debian

À nouveau, le terminal *xfce4-terminal* est recommandé. Le paquet *fonts-noto-color-emoji*.

Toutefois, un problème reste avec l'écureuil. Sous Ubuntu et Debian, le caractère écureuil existe déjà, mais ne s'affiche pas proprement. On peut appliquer un patch qui permet d'afficher les émojis correctement dans son terminal. Pour cela, il suffit de faire :

```
ln -s $PWD/debian/75-fix-squirrel-emojis.conf /etc/fonts/conf.avail/75-fix-squirrel-
↪emojis.conf
ln -s /etc/fonts/conf.avail/75-fix-squirrel-emojis.conf /etc/fonts/conf.d/75-fix-
↪squirrel-emojis.conf
```

Après redémarrage du terminal, l'écureuil devrait s'afficher correctement.

Pour supprimer le patch :

```
rm /etc/fonts/conf.d/75-fix-squirrel-emojis.conf
```

À noter que ce patch est inclus dans le paquet Debian.