

Polytech City

Table des Matières

Règles du jeu	2
But	2
Début	2
Bâtiments	2
Taux de Satisfaction	4
Taux d'Imposition	4
Évènements	4
Structure du code	5
Programme principal	5
dessin.py	5
event.py	5
menu.py	5
tools.py	5
Addons	6
Le menu	6
Plusieurs actions par tour	6
Interface active	6
Répartition du travail	7
Difficultés rencontrées	7

Règles du jeu

Polytech City est un jeu codé sous Python basé sur le jeu SimCity. Le code est basé sur le module turtle permettant d'ajouter du graphisme à Python.

But

Le joueur est Maire d'une toute nouvelle ville et peut construire, détruire plusieurs types de bâtiments. Le but du jeu est d'avoir le plus d'habitants possibles en évitant la faillite de votre ville et il faut également que vos habitants soient satisfaits de votre gestion de la ville.

Le score correspond au nombre d'habitants de la ville et il existe un taux de satisfaction permettant de voir comment le joueur gère sa ville.

Le score augmente en fonction des bâtiments construits pendant le jeu.

Début

Le joueur commence avec un budget initial de 200 \$ et peut commencer à construire sa ville. Dès le début, le joueur peut construire une maison ou démolir une construction. Puis au fur à mesure de l'avancement de sa ville, le joueur débloque d'autres types de bâtiments. Le jeu se déroule au tour par tour, le joueur peut effectuer 4 actions par année.



Bâtiments

- Habitation

La maison est le premier bâtiment que le joueur peut construire.

L'immeuble est débloqué à partir de 30 habitants.



Tableau explicatif :

	Dessin	Construction	Démolition	Habitants	Besoin
Maison		30 \$	10 \$	0-5	
Immeuble		60 \$	20 \$	0-50	30 habitants

*Remarque : Le coût de démolition correspond au tiers du coût de construction
Pour la plupart des bâtiments suivants, il faut 5 habitants dans la ville pour les débloquer*

- Commerce

Il existe deux types de commerce : l'épicerie et l'usine. Les habitants ont besoin d'usine pour travailler et l'épicerie pour se ravitailler.








	Dessin	Construction	Démolition	Besoin	Taux de Satisfaction
Epicerie		45 \$	15 \$	1/50 habitants	2%
Usine		75 \$	25 \$	1/30 habitants	5%

- Services

Beaucoup de services sont disponibles pour la ville : Police, Pompier, Hôpital, Ecole, Mairie et Parc. Plus votre ville sera grande, plus vos habitants auront besoin de ces services. Certains services auront différents impacts sur la ville :

Le parc augmente le taux de satisfaction des habitants mais coûte cher ! Si celui est détruit le taux de satisfaction baisse.

Les pompiers ou les hôpitaux doivent être entretenus chaque année, des frais d'entretiens annuels sont donc facturés (Cf. tableau pour plus de détails)

	Dessin	Construction	Démolition	Conditions	Besoin	Taux de Satisfaction	Frais d'entretien
Police		75 \$	25 \$	5 habitants	1/50 habitants		
Pompier		120 \$	40 \$	5 habitants	1/50 habitants		40 \$ /année
Hôpital		120 \$	40 \$	5 habitants	1/50 habitants		40 \$ /année
Ecole		210 \$	70 \$	50 habitants	1/50 habitants		70 \$ /année
Mairie		210 \$	70 \$	100 habitants		15%	
Parc		360 \$	120 \$	20 habitants	1/60 habitants	20% (Si détruit -15 %)	120 \$ /année
Stade		1500 \$	500 \$	150 habitants		15%	500 \$ / année

Taux de Satisfaction

Par défaut, le taux de Satisfaction baisse de 1 % chaque année. Et si la ville ne contient pas assez d'un certain type de bâtiment, la baisse augmente de 1 % par bâtiment manquant (Cf. colonne Besoin du tableau).

Taux d'Imposition

Le taux d'Imposition est la source de revenu du joueur. Par défaut, il est fixé à 10%. Il pourra être modifié lorsque la Mairie sera construite. Attention, lors de la modification du taux d'imposition, les revenus vont augmenter tandis que le taux de Satisfaction lui diminuera.

Évènements

Des évènements positifs et négatifs toucheront votre ville de façon aléatoire dans le jeu. Voici une liste non exhaustive : JO, la plus belle ville, catastrophe naturelle, incendie, épidémie. Les chances d'apparition d'un évènement négatif sont réduites par la construction de bâtiments spéciaux telles que les pompiers pour l'incendie ou encore le poste de police pour le cambriolage.

Structure du code

Le code est divisé par 4 modules qui interagissent entre eux et d'un code principal (`polytechCity.py`)
Le programme principal est constitué d'une boucle qui effectue des actions à chaque tour du joueur, et pendant laquelle elle va appeler des fonctions des différents modules.

Programme principal

Tout d'abord, on initialise toutes les variables et la configuration du plateau. On rentre ensuite dans « la boucle de jeu ». Une fois dans la boucle, il y a une gestion de plusieurs « sous-menu » améliorant l'ergonomie du jeu.

`dessin.py`

Comme son nom l'indique, ce module regroupe toutes les fonctions graphiques du code. De la plus utilisée (*dessineRectangle*) à la moins utile (*dessineGrille*).

`event.py`

Ce module gère en effet les événements du jeu. On peut y retrouver toutes les fonctions pour les différents événements (JO, tornade, séisme, épidémie etc.). Ces événements peuvent agir sur le nombre d'habitants, la configuration du plateau, l'argent et d'autres paramètres.

`menu.py`

C'est un module contenant une seule fonction qui est agir. Cette fonction un peu longue avec beaucoup de vérification permet de construire (ou démolir) une case du plateau. Elle est appelée dans le menu.

`tools.py`

Ce sont différents outils utiles pour le reste du code. On y retrouve notamment, le calcul du taux de satisfaction, l'aléatoire du nombre d'habitants pour un bâtiment, le nombre d'un certain type de bâtiment.

Add-ons

Le menu

Il est amélioré par rapport à celui par défaut. Il est un peu plus « intelligent ». En effet, dès le début le joueur n'a pas beaucoup de possibilités mais au fur à et mesure il débloquent certains bâtiments grâce à une liste contenant les bâtiments débloqués, le menu gère cela. Egalement, le menu s'adapte à « l'échec » du joueur (grâce à une variable *fail*) et évite alors de passer un tour pour rien, notamment lorsque les coordonnées ne sont pas bonnes ou alors qu'il existe déjà un bâtiment sur ces coordonnées. Et enfin, la gestion du « Précédent » sur le même principe que plus haut.

Plusieurs actions par tour

Le joueur peut effectuer 4 actions par tour avant de passer à l'année suivante. Il a suffi de rajouter une boucle pour faire ses actions. Sinon le jeu était beaucoup trop rapide et la gestion du jeu était très différente.

Interface active

Pour alerter le joueur d'un manque d'un certain type de bâtiment, cela est plus simple de le voir graphiquement que dans la console. C'est pourquoi il a fallu trouvé un moyen de le faire apparaître sur le plateau en sachant s'il manquait un bâtiment ou non. Nous avons donc utilisé la fonction du calcul du taux de satisfaction et ajouté un paramètre booléen « dessin ». Cela permet donc d'actualiser chaque tour sans calcul du taux de satisfaction. De plus, cette fonction est appelée chaque année pour un nouveau taux avec le paramètre « dessin » false par défaut. Par cette méthode, on évite de réécrire une fonction faisant plus ou moins la même chose et si l'on veut dessiner, cela ne change pas le taux de satisfaction.

Répartition du travail

- dessin.py : Module géré par STROBBE Nathan
 - event.py : Module géré par SCIARRA Aurélien
 - menu.py : Module codé par le binôme
 - tools.py : Satisfaction géré par Aurélien et Nathan puis modifié par la suite par Nathan pour gérer l'interface graphique (paramètre dessin)
- Autre outils géré par Aurélien
- polytechCity.py : Code modifié en grande partie par Nathan, liens avec les modules gérés par Aurélien

Difficultés rencontrées

- Module graphique au début (géré par Nathan)
J'ai eu quelques soucis pour dessiner les différents bâtiments mais j'ai su vite les régler en pensant à faire plusieurs fonctions telle que *dessineRectangle()*. Et j'ai fait des dessins assez simples pour la représentation de chaque bâtiment.
- Event JO (géré par Aurélien)
J'ai eu un problème dû à l'unicité de l'évènement, réglé grâce à une variable booléenne.
J'ai également découpé la fonction en deux pour simplifier le code et pour renvoyer deux valeurs différentes (la liste JO et le gain)
- Interface graphique (géré par Nathan)
J'ai dû utilisé une fonction déjà présente dans le code qui calcule le taux de satisfaction. En effet pour éviter de répéter à nouveau la vérification de chaque bâtiment manquant, j'ai ajouté un paramètre dessin (à False par défaut) et grâce à un if, la fonction permet soit de renvoyer la valeur de satisfaction soit de dessiner l'interface.