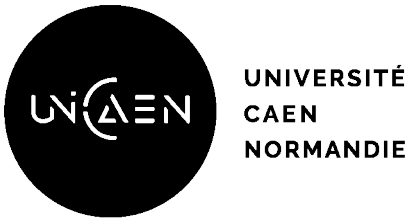


Université de Caen
Licence Informatique
2017/2018

Projet Conception Logiciel Semestre 2
Rapport

Par Chaid Akacem Jasmine
et Hugo Baudin

Sommaire

Partie I

Présentation du projet

Dans le cadre de notre formation de Licence Informatique, il nous a été demandé de réaliser un projet en langage python en 40h, le choix du sujet étant libre parmi 6 sujets. Nous allons vous présenter ce qui a été fait durant ce projet.

A Sujet

Parmi les 6 sujets, trois nous ont attiré. Nous avons hésité entre le Puzzle Quest, le Manic Shooter et l'Éditeur de livre dont vous êtes le héros. Afin de choisir, nous avons écrit toutes les idées de fonctionnalités ainsi que d'améliorations futures que nous avions sur chaque sujet. Étant donné que nous avions plus de propositions pour l'éditeur de livre, nous nous sommes dirigés sur ce projet.

Nous avons conçu un éditeur de livre qui permet la visualisation du graphe du livre, avec au maximum 54 paragraphes dans le livre, avec un système de statut pour bien identifier les paragraphes, le tout sauvegardé au sein d'un « livre objet ». Nous avons aussi conu le logiciel de lecture associé aux fichiers crées par l'éditeur, permettant de visualiser le graphe et son chemin à la fin du livre.

B Ce qui existe déjà

Il existe déjà des logiciels pour créer des livres dont vous êtes le héros : Advelh, qui est un logiciel existant depuis 1995 et disponible en français, InkleWriter ou encore Twine.
Ainsi que des logiciels pour y jouer, qui sont beaucoup plus populaires, comme le jeu Hand of Fate, par les studios de Defiant Development en 2015.
Il existe également des sites web pour proposer et lire des histoires dont vous êtes le héros, comme la section interactives de writing.com.

Partie II

Fonctionnalités

A Création d'objets

Nous allons détailler les fonctionnalités de création des différents objets utilisés au sein de notre éditeur de livre.

a Création de paragraphes

La création de paragraphe passe par une fenêtre séparée de la fenêtre principale. Il y a trois étapes :

1. Écriture du paragraphe
2. Création de l'objet paragraphe
3. Affichage du paragraphe sur le graphe

L'utilisateur va renseigner le titre de son paragraphe, ce qui lui permettra de l'identifier par la suite dans la fenêtre principale. Puis, l'utilisateur tape son texte. Enfin, il doit renseigner le statut de son paragraphe, si c'est un paragraphe de début de livre, de milieu de livre ou alors de fin. Il y a deux statuts pour la fin, fin gagnante ou fin perdante. La fin gagnante correspond à une fin jugée bonne, la fin perdante à un échec dans l'histoire.
La création de l'objet se fait lorsque l'utilisateur valide les informations rentrées. L'objet paragraphe est instancié et les informations titre, contenu et statut renseignées dans les attributs correspondant. La distinction entre fin gagnante et fin perdante se fait au niveau d'un attribut séparé du statut. Si le statut n'a pas été renseigné, un statut par défaut est utilisé.
Enfin, le paragraphe devient visible sur la fenêtre principale et identifiable avec son titre. Une commande lui est associé, afin de pouvoir ouvrir une fenêtre de gestion, permettant quelques modifications.

b Création de choix

La création d'un choix nécessite la précision du paragraphe dans lequel il doit être crée. Une fois ceci fait, l'utilisateur doit renseigner, dans une fenêtre séparée, le titre de son choix et quelques phrases pour son contenu, si besoin. Ensuite, l'objet choix est instancié et le titre ainsi que le contenu sont stockés au sein d'attributs spécifiques. L'objet choix garde une référence de son « parent » dans un attribut. L'objet paragraphe garde également une référence à tout ses choix dans un attribut particulier.

c Création de liens

Le lien est un objet simple à comprendre, mais complexe à créer.
Un lien va d'un paragraphe dit paragraphe_I, pour paragraphe Input, gràce à un choix, à un autre paragraphe dit paragraphe_O, pour paragraphe Output. L'utilisateur doit renseigner ces trois objets, avant l'objet lien puisse être crée.
De plus, les deux paragraphes gardent une référence à leurs liens. Le P_I garde une trace du lien dans un attribut

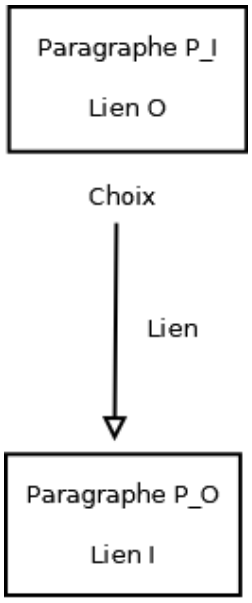


FIGURE II.1 – Schéma explicatif de la structure d'un lien

Lien O, pour lien output, ce qui signifie ici lien sortant. Le P_O garde une trace du lien dans un attribut Lien I, pour lien input, ce qui signifie ici lien entrant. Enfin, le choix est lié au lien avec un attribut.

B Suppression d'objets

Nous allons décrire les fonctionnalités permettant de supprimer les objets précedemment créés.

a Suppression de liens

La suppression de liens se fait par un menu spécifique. Cette fonction permet à l'utilisateur de supprimer les différents liens entre paragraphes.
Pour identifier le lien à supprimer, l'algorithme demande à l'utilisateur de préciser de quel paragraphe part le lien, puis de quel choix part le lien. Ainsi, l'algorithme peut facilement identifier quel lien il doit supprimer.
La fonction supprimera toutes les référence au lien dans les différents objets, donc dans deux paragraphes, dans le choix et dans l'objet histoire. Enfin, on efface l'affichage du lien sur la fenêtre en utilisant ses coordonnées pour le repérer.

La fonction réagit de deux manières différentes en fonctions du mode de suppression. En effet, les liens peuvent être effacés de façon direct, c'est-à-dire en supprimant le lien, ou de façon indirect, c'est-à-dire lorsque l'utilisateur supprime un objet choix qui est lié à un lien.
De plus, lors de la suppression d'un paragraphe, les liens qui arrivent vers ce paragraphe ainsi que ceux qui sortent de ce paragraphe doivent être effacé. L'algorithme prends en compte cette particularité de suppression lorsqu'il est appelé avec l'algorithme de suppression de paragraphe.

b Suppression de choix

De même que pour les liens, les choix se supprime par le même menu que la suppression des liens.
La fonction agit également de deux manières selon le mode de suppression. Si l'on utilise l'algorithme directement, cela se fait de manière directe. Si l'on supprime un paragraphe, cela supprimera le choix de manière indirecte.

Lors de la suppression direct, on récupère d'abord le choix avec le paragraphe dans lequel il se trouve. Puis, sa référence est supprimé de tout les objets où il apparait. Par ailleurs, si un lien à été lié au choix, ce lien est supprimé de façon indirecte grâce à l'algorithme ci dessus.

c Suppression de paragraphes

Au contraire des suppression précédente, la suppression des paragraphes se réalise en cliquant sur un bouton situé dans une fenêtre spécifique au paragraphe.
Lorsque l'on clique sur le bouton, l'algorithme commence par effacer les coordonnées du paragraphes afin de pouvoir les réutiliser pour un nouveau paragraphe, puis on supprime le référencement du paragraphe dans l'objet Histoire. Ensuite, les choix liés au paragraphes sont supprimés avec l'algorithme précédent. Cela permet de supprimer les liens sortant de ce paragraphe. Pour supprimer les liens qui mènent vers ce paragraphe, il faut appeler l'algorithme de suppression des liens dans ce cas précis.
Enfin, l'algorithme supprime le paragraphe de la fenêtre.

C Visualisation du graphe

Le graphe de l'éditeur de livre est au centre de la fenêtre. Lors de la création de paragraphe, celui-ci se place sur une ligne spécifiée en fonction de son statut. La première ligne est réservée pour les paragraphes de début. Les six lignes qui suivent servent à positionner les paragraphes de milieu. Enfin, les deux dernières lignes est gardée pour les paragraphes de fin. Chaque ligne peut accueillir six paragraphes, ce qui explique la limite de 54 paragraphes pour notre éditeur. Dans le graphe, seul le titre du paragraphe est affiché.
Le contenu et le titre de chaque paragraphe restent accessibles et modifiables même placé dans le graphe, il suffit d'enclencher une commande spécifique afin de faire afficher une fenêtre réservée à la gestion du paragraphe en question.
Les choix ne sont pas visibles sur le graphe, étant uniquement des contenus, mais ils peuvent toujours être consulté via la fenêtre de gestion des paragraphes.
Les liens terminent de former le graphe. Ils relient les paragraphes avec des ligne se terminant par une flèche.

D Lecture

Le mode Lecture se découpe en quatre fenêtres différentes.
La première est une fenêtre de démarrage qui demande à l'utilisateur s'il souhaite quitter le mode Lecture ou charger un livre.
La deuxième fenêtre permet au joueur de charger son livre en le repérant avec son nom.
La fenêtre principale du mode Lecture s'ouvre. Elle est divisée en deux, comme un livre. A gauche se trouve le texte du paragraphe en cours, et à droite sont placés les choix possibles. Lorsque le joueur appuie sur un de ces choix, le mode lecture capte le paragraphe qui suit et l'affiche à l'écran, avec les choix possibles.

Lorsque le joueur arrive sur un paragraphe de fin, la page de droite affiche un bouton spécifique, qui dépend du statut de la fin (gagnante ou perdante) permettant de fermer la fenêtre de jeu. Une fenêtre de fin s'ouvre, affichant le graphe ainsi que le chemin parcouru durant le jeu en soulignant les paragraphes lus et les liens entre ces paragraphes. Le mode lecture renseigne également le joueur sur le nombre d'étapes qu'il lui a fallu pour arriver à une fin.
Enfin, une fois que le joueur quitte la fenêtre du graphe, le mode lecture propose au joueur si il veut recommencer le livre.

E Organisation du projet

L'organisation du projet a été assez compliqué au sein de notre groupe. Nous étions trois au départ, et lors du commencement des objets, nous nous sommes répartis les trois objets principaux, à savoir paragraphe, liens et choix. Puis nous n'avions plus de nouvelles, plus de présence aux TP's, nous ne savions pas si il allait revenir. Jasmine s'est occupée des liens, de l'objet histoire et des choix, d'abord commencé par Hugo, mais il a dû écrire complétement la classe paragraphe, presque peu écrite par notre ancien collègue.

Une autre personne nous a rejoint, quand nous pensions encore que nous étions quatre. Nous lui avons laissé plusieurs heures pour comprendre notre sujet, le peu que nous avions écrit tout en lui expliquant. Ensuite, nous lui avons donné l'affichage des paragraphes à coder, selon un résultat bien précis, tandis que Jasmine s'occupait de la création des liens, et Hugo de continuer la classe choix. Malheureusement, cette personne a quitté sans prévenir et sans explication notre groupe, sans avoir codé quoi que ce soit.

Nous avons mis du temps á comprendre que nous allions rester á deux, sans aucune information pendant presque 20h de projet. Cela a entrainé moins de communication dans le groupe pendant cette période. Hugo s'est occupé de tout ce qui avait un rapport direct avec les paragraphes, en particulier leur création, leur affichage dans la fenêtre et la possibilité d'afficher et de modifier le contenu du paragraphe. Jasmine s'est occupée de la gestion de la fenêtre principale, de la création des liens et des choix.
Après cette période, nous avons travaillé beaucoup plus ensemble afin de lier la création de tout les objets et leur affichage. Jasmine a travaillé sur un moyen de sauvegarder l'histoire, la suppression des liens, des choix et Hugo sur la manière de vérifier que l'histoire était valide pour la lecture.
Pour le mode lecture, Jasmine a travaillé sur la fenêtre de début, le chargement de l'histoire dans le logiciel, la fin avec le dessin du graphe avec le chemin pris par le joueur ainsi que la possibilité de recommencer. Hugo a travaillé sur le choix d'un paragraphe de début dans l'histoire, le déroulement du jeu, l'affichage du texte et la possibilité d'enclencher un choix.

Partie III

Elèments techniques

A Algorithmes

a Algorithme de Création des liens

Nous estimons que la création des liens est l'étape la plus importante dans la réalisation d'un livre dont vous êtes le héros. De plus, l'algorithme fait appel à toute les classes d'objet instanciés.
L'utilisateur appuie sur la commande « Créer un lien », ce qui ouvre une nouvelle fenêtre. Une fonction est immédiatemment appelée, la fonction *remplir_listbox_p*. Cette fonction permet de compléter une listbox de la fenêtre avec les titres de tout les paragraphes qui ont été créé. Cette listbox permet à l'utilisateur de choisir un paragraphe, en l'occurence, celui dont va partir le lien, avec une contrainte : un paragraphe ayant pour statut « Fin » ne peut être un paragraphe input (voir Schéma explicatif de la structure d'un lien) . La fonction *remplir_listbox_p* prends en compte cette subtilité. Ensuite, l'algorithme attend une réponse de l'utilisateur pour continuer.
Une fois le premier paragraphe choisi, la fonction *get_p* est appelée pour retrouver l'objet paragraphe à partir de son attribut titre. Puis, on teste si le paragraphe contient au moins un choix non lié. C'est essentiel car un lien doit partir d'un choix. Si le paragraphe ne contient aucun choix, une erreur est levée. Une fenêtre informant l'utilisateur de son erreur s'affiche et le programme s'interrompt.
Si le paragraphe possède au moins un choix libre, une nouvelle listbox va se créer, et la fonction *remplir_listbox_c* est appelée, afin de remplir la listbox avec les noms des choix du paragraphe sélectionné. Puis, l'utilisateur doit choisir parmi cette liste le choix qu'il veut associer à son nouveau lien.
Lorsque l'utilisateur a sélectionné son choix, le programme retrouve le choix à partir de son attribut nom grâce à la fonction *get_c*. Il reste à renseigner le deuxième paragraphe, celui où va aboutir le lien. Une nouvelle listbox est crée, et la fonction *remplir_listbox_p* est de nouveau appellée, avec cette fois deux contraintes :

1. Un paragraphe de statut « Début » ne peut être un paragraphe output (voir Schéma explicatif de la structure d'un lien)
2. Le premier paragraphe sélectionné ne peut pas être à nouveau sélectionné

Une fois ces contraintes respectées, l'algorithme attends que l'utilisateur choisisse le dernier paragraphe.
Lorsque l'algorithme possède tout les élèment nécessaires à la création du lien, il commence le traitement des objets. Tout d'abord, le lien est généré grâce à la fonction *generator_l*, qui instance l'objet lien et le rajoute immédiatemment à l'attribut liens de l'objet histoire. Puis ses attributs sont mis à jour avec une méthode lien. L'attribut position est mis à jour grâce à une méthode à part, il prends les positions des deux paragraphes qu'il relie. Les autres objets sélectionnés par l'utilisateur doivent eux aussi subir une mise à jour.

- Le paragraphe input pour ajouter le lien à son attribut lien_O
- Le paragraphe output pour ajouter le lien à son attribut lien_I
- Le choix pour le lier au lien

Enfin, le lien est représenté par une ligne sur le canvas.

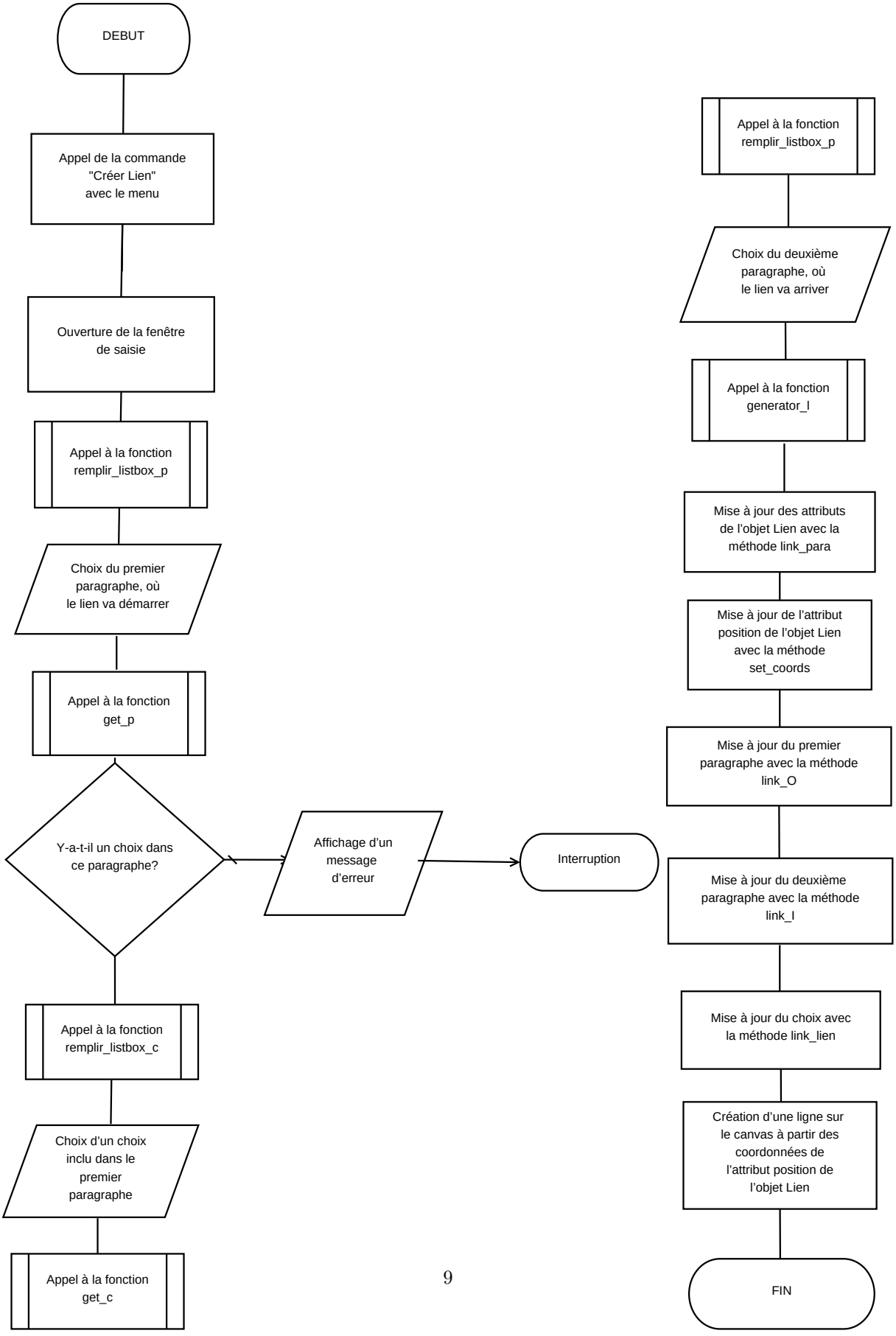

DEBUT
Appel de la commande "Créer Lien" avec le menu
Ouverture de la fenêtre de saisie
Appel à la fonction remplir_listbox_p
Choix du premier paragraphe, où le lien va démarrer
Appel à la fonction get_p
Y-a-t-il un choix dans ce paragraphe?
Affichage d'un message d'erreur
Interruption
Appel à la fonction remplir_listbox_c
Choix d'un choix inclu dans le premier paragraphe
Appel à la fonction get_c
Appel à la fonction remplir_listbox_p
Choix du deuxième paragraphe, où le lien va arriver
Appel à la fonction generator_l
Mise à jour des attributs de l'objet Lien avec la méthode link_para
Mise à jour de l'attribut position de l'objet Lien avec la méthode set_coords
Mise à jour du premier paragraphe avec la méthode link_O
Mise à jour du deuxième paragraphe avec la méthode link_I
Mise à jour du choix avec la méthode link_lien
Création d'une ligne sur le canvas à partir des coordonnées de l'attribut position de l'objet Lien
FIN

FIGURE III.1 – Algorithme de création des liens

b Algorithme de création de paragraphe

L'utilisateur qui souhaite créer un paragraphe doit cliquer sur la commande « Crér un paragraphe ». Cela fait appel à la fonction *EcrirePara*, qui ouvre une fenêtre possédant des champs pour le titre et le contenu du paragraphe, ainsi qu'une liste de statut à séléctionner.
Lorsque tout les champs sont remplis, l'utilisateur appuie sur le bouton « Valider », qui appelle la fonction *CreerPara*. La fonction va commencer par stocker le titre, le contenu et le statut dans leur attribut correspondant de la classe Paragraphe. Si aucun statut n'a été choisi, celui de « Milieu » sera donné par défaut. On fait ensuit appel á la fonction *coord_pour_p* afin de donner des coordonnées au paragraphe en fonction de son statut. Une fois les coordonnées prises, si l'attribut position vaut -1, cela veut dire qu'il n'y a plus de coordonnées disponibles pour placer le paragraphe sur la fenêtre. La création est avorté et renvoie un message d'erreur. Les paragraphes de statut Fin Gagnante et Fin Perdante prennent le statut de Fin afin de faciliter leur traitement dans les autres fichiers du logiciel et l'attribut fin_g prend True si c'est une Fin Gagnante, sinon False.
Enfin, le bouton de paragraphe se place dans le canevas selon son attribut position. Si c'est un paragraphe de statut fin, il prend la couleur verte ou rouge, pour fin gagnante ou fin perdante.

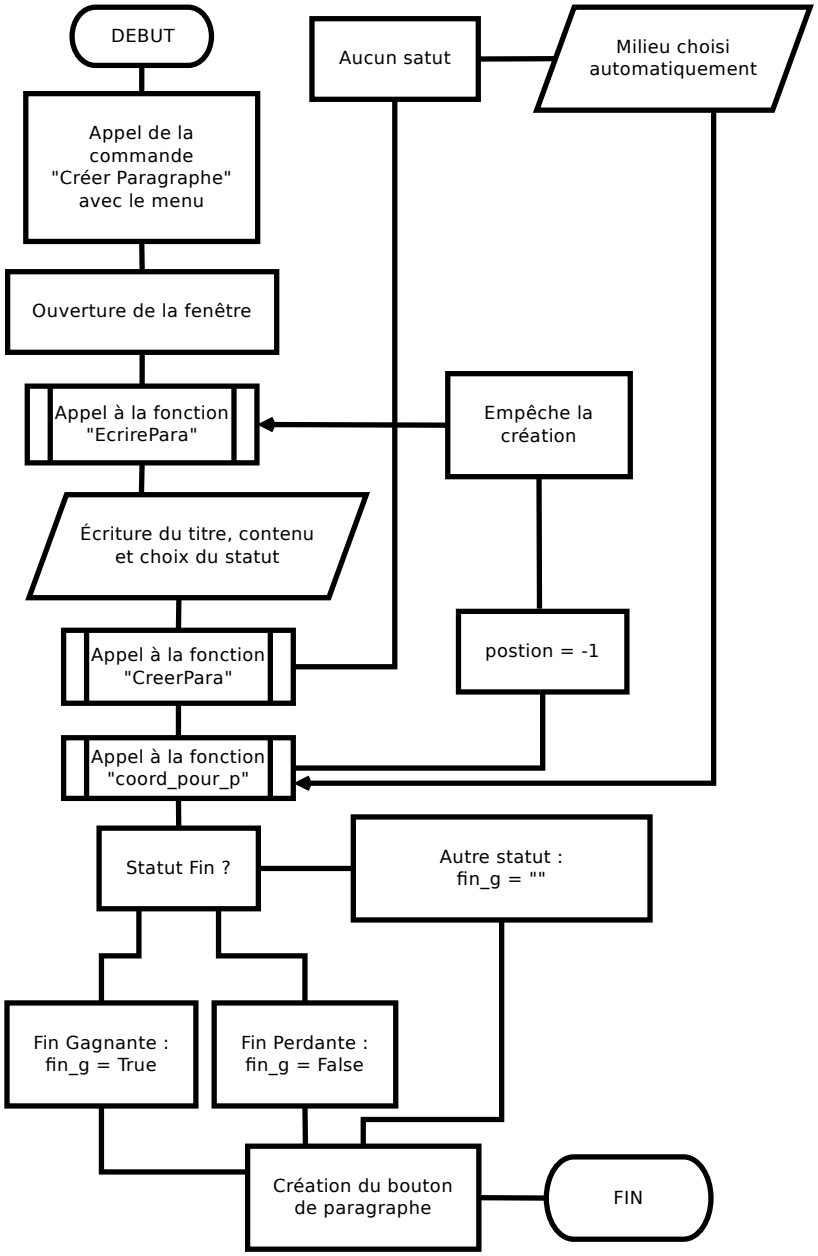


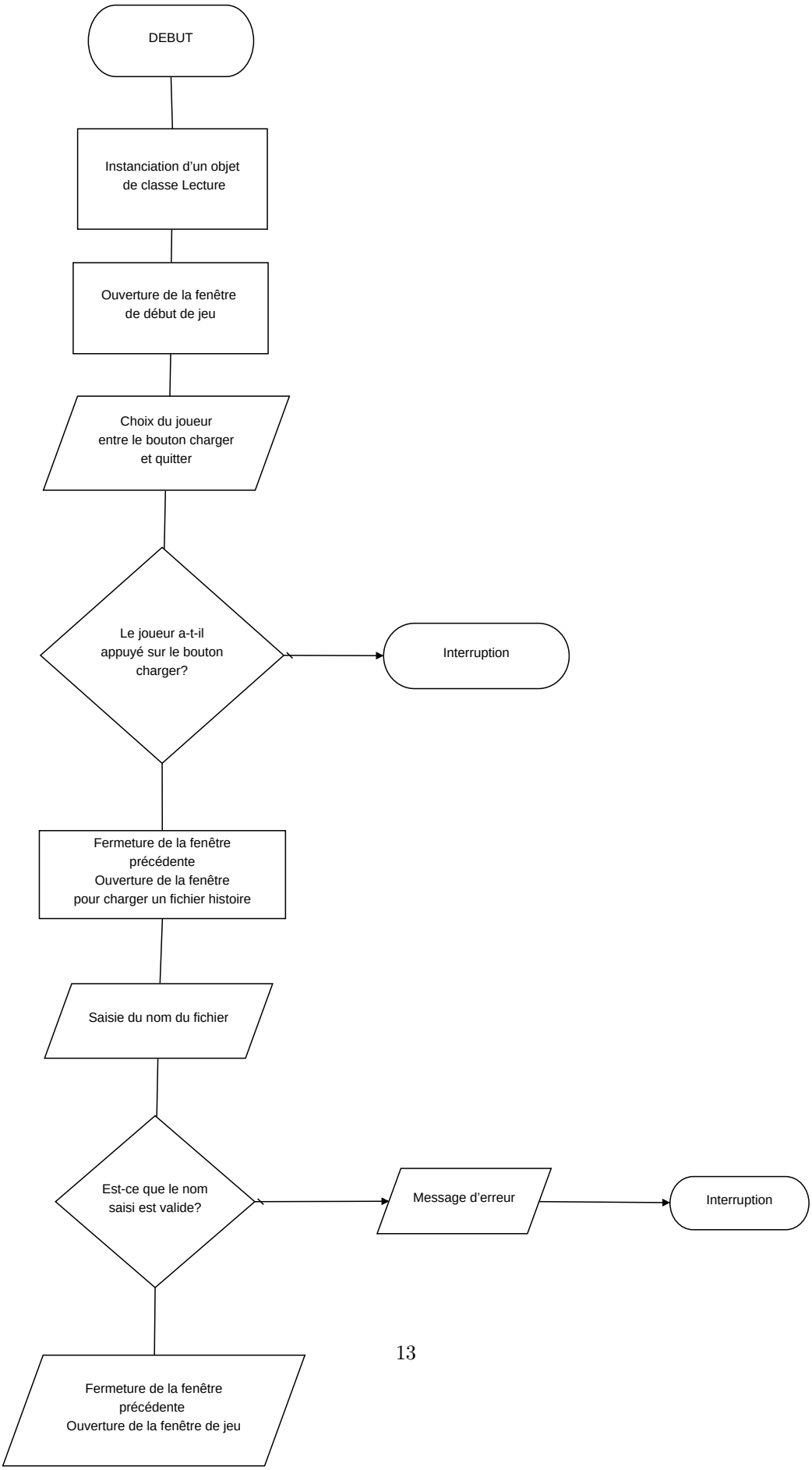
FIGURE III.2 – Algorithme de création de paragraphe

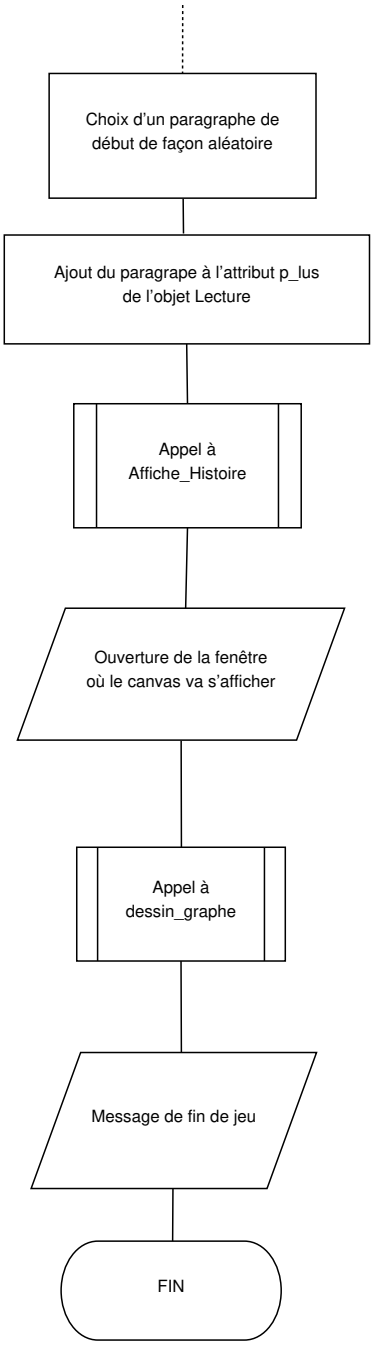
c Mode Lecture

Algorithme général

Le fonctionnement général du mode lecture peut être assimilé au déroulement d'un jeu. Le mode lecture fait appel à une classe spécifique : la classe Lecture.
Lorsque le mode lecture est lancé, une première fenêtre s'ouvre, la fenêtre de début de jeu. Le joueur peut commencer le jeu en appuyant sur Charger, raccourci pour la commande « Charger un fichier histoire »,ou quitter le jeu en appuyant sur Quitter. Si le joueur n'appuie pas sur le bouton Charger, c'est qu'il a appuyé sur le bouton Quitter. La fenêtre se ferme et le programme s'interrompt.
Si le joueur a appyé sur le bouton Charger, la fenêtre actuelle se ferme, pour ouvrir une autre fenêtre. Cette fenêtre permet au joueur de saisir le nom du fichier dans lequel le livre est sauvegaré. Puis le programme évalue sur le nom du fichier est valide, c'est à dire si il correspond à un fichier existant dans le dossier et si il peut l'ouvrir. Dans le cas où le nom rentré n'est pas valide, une erreur est levée, montrée à l'utilisateur sous la forme d'un message, puis le programme s'interrompt.
Si le nom du fichier est valide, le fichier histoire sauvegardé dans ce fichier est récupéré. Le programme cherche dans l'attribut paragraphe le ou les paragraphes de début présents. Si il y en a plusieurs, il en choisit un aléatoirement. De plus, ce paragraphe est ajouté à l'attribut p_lus de l'objet lecture. C'est essentiel car l'attribut p_lus représente tout les paragraphes lus par le joueur, le premier compte, si jamais il y a plusieurs début possible.
Une fonction particulière est appelée, la fonction *Affiche_Histoire*. C'est une fonction qui est tout le temps rappelée au cours du jeu, car elle sert à afficher les paragraphes ainsi que les choix. (voir figure)
Lorsque *Affiche_Histoire* n'est plus appelée, la fenêtre de jeu se ferme, laissant place à la fenêtre où se dessinera le graphe, grâce à la fonction *dessin_graphe*. Enfin, un message de fin de jeu s'affiche à l'attention du joueur, pour lui confirmer la fin du jeu.
Une fonctionnalité supplémentaire a été implantée. À la fin du jeu, le joueur peut choisir de rejouer au même livre. Elle relance le programme au niveau du tirage aléatoire du paragraphe de début.

FIGURE III.3 – Algorithme général du mode lecture


DEBUT
Instanciation d'un objet de classe Lecture
Ouverture de la fenêtre de début de jeu
Choix du joueur entre le bouton charger et quitter
Le joueur a-t-il appuyé sur le bouton charger?
Interruption
Fermeture de la fenêtre précédente
Ouverture de la fenêtre pour charger un fichier histoire
Saisie du nom du fichier
Est-ce que le nom saisi est valide?
Message d'erreur
Interruption
Fermeture de la fenêtre précédente
Ouverture de la fenêtre de jeu


Choix d'un paragraphe de début de façon aléatoire
Ajout du paragrape à l'attribut p_lus de l'objet Lecture
Appel à Affiche_Histoire
Ouverture de la fenêtre où le canvas va s'afficher
Appel à dessin_graphe
Message de fin de jeu
FIN

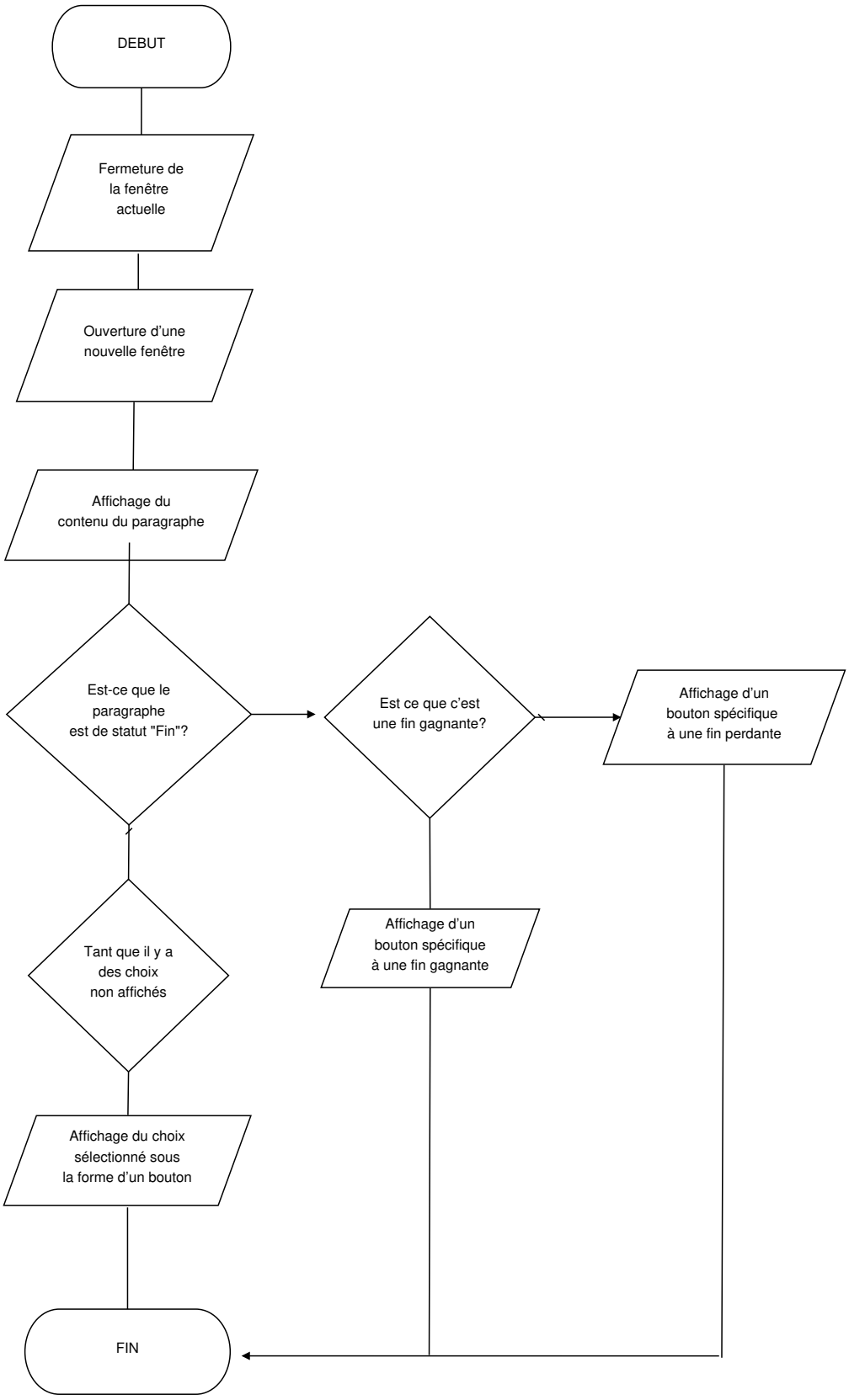
Algorithme de Affiche_Histoire

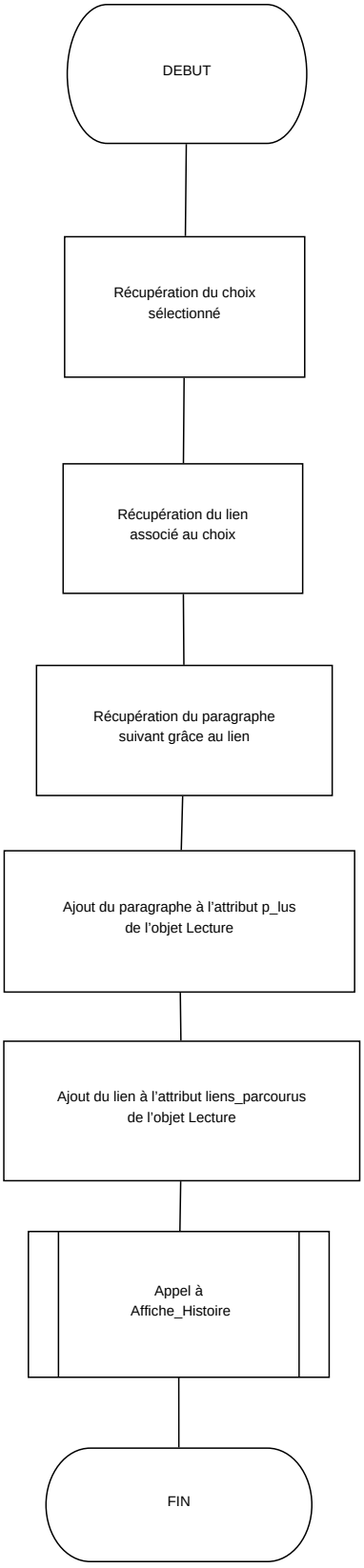
Pour comprendre le mode lecture, il nous semble essentiel de décrire le fonctionnement de *Affiche_Histoire*, qui explique le déroulement du jeu.
Le programme commence par fermer la fenêtre courante, pour la remplacer par une nouvelle fenêtre vierge. Le contenu du paragraphe actuel est affiché. Le traitement est différent selon si le paragraphe est de statut fin ou non. Si le paragraphe est de statut fin, le programme cherche si c'est une fin gagnante ou perdante et fait afficher le bouton adéquat dans la fenêtre. Ces boutons entraînent la fin de l'algorithme lorsque le joueur clique dessus. C'est normal, c'est la fin du livre.
Si le paragraphe n'est pas un paragraphe de fin, alors le programme boucle sur les choix du paragraphes, afin d'afficher leur contenu sous forme d'un bouton de type poussoir, afin que le joueur ne puisse en choisir qu'un seul. Cet ensemble de boutons possède la même commande : *Sélection*. Cette fonction permet d'enclencher de nouveau *Affiche_Histoire*, mais avec des traitements d'objets.

Algorithme de Sélection

Sélection est une fonction appellée par une commande : dès que le joueur appuie sur un choix, *Sélection* s'exécute. La fonction récupère l'objet choix sélectionné par le joueur, et trouve ainsi le lien associé au choix. Puis, grâce au lien, le programme retrouve le paragraphe suivant. Ces deux derniers objets sont ajoutés respectivement à l'attribut liens_parcourus et et p_lus de l'objet lecture, afin qu'il se souvienne du tracé du joueur. Cela sera essentiel pour tracer le chemin complet à la fin du jeu.
A la fin de ce traitement d'objets, *Sélection* rappelle la fonction*Affiche_Histoire* avec le nouveau paragraphe, afin de continuer le livre.

FIGURE III.4 – Algorithme de Affiche_Histoire et de Sélection


DEBUT
Fermeture de la fenêtre actuelle
Ouverture d'une nouvelle fenêtre
Affichage du contenu du paragraphe
Est-ce que le paragraphe est de statut "Fin"?
Est ce que c'est une fin gagnante?
Affichage d'un bouton spécifique à une fin perdante
Tant que il y a des choix non affichés
Affichage d'un bouton spécifique à une fin gagnante
Affichage du choix sélectionné sous la forme d'un bouton
FIN


DEBUT
Récupération du choix sélectionné
Récupération du lien associé au choix
Récupération du paragraphe suivant grâce au lien
Ajout du paragraphe à l'attribut p_lus de l'objet Lecture
Ajout du lien à l'attribut liens_parcourus de l'objet Lecture
Appel à Affiche_Histoire
FIN

d Dessin du graphe

Le dessin du graphe à l'issue du mode lecture fait le lien réel entre l'éditeur et le mode lecture. Il permet d'afficher le chemin parcourus au joueur, au moyen de couleurs, à la fin du jeu, et de lui laisser apprécier les autres chemins qu'il aurait pû emprunter pour finir le livre, sans toutefois lui donner la solution. Le plaisir d'un livre dont vous êtes le héros repose sur la répétition du jeu, le fait de choisir d'autres chemins, et de chercher à tout prix comment accéder à la « bonne fin » (selon le joueur, elle peut être différente pour l'utilisateur). De plus, cette fonction pourrait être également implanté pour l'Éditeur de livre, ce qui permettrait à l'utilisateur de reprendre un livre en cours.
Le programme commence par ouvrir la nouvelle fenêtre et mettre en place le canevas, abrégé en canvas (terme anglophone) dans l'algorigramme.
Ensuite, le programme parcourt tout les paragraphes contenus dans l'attribut paragraphe de l'objet Histoire, qui lui même est contenu dans un attribut de l'objet Lecture. Le programme évalue si le paragraphe qu'il sélectionne a été parcouru, c'est à dire si il est dans l'attribut p_lus, pour paragraphes lus, de l'objet Lecture.
Si oui, il est représenté sur le canevas par un bouton coloré. Sinon, il est représenté comme un bouton normal.
Après avoir parcouru les paragraphes, le programme parcourt ensuite les liens contenus dans l'attribut liens de l'objet Histoire. Si ils ont été parcourus, ils sont contenus dans un attribut de l'objet Lecture, nommé liens_parcourus. Il suffit au programme de tester si le lien sélectionné est présent dans l'attribut liens_parcourus.
Si oui, le lien est représenté comme une flèche colorée. Sinon, le lien est représenté sous la forme d'une flèche classique, de couleur noire.
A la fin de l'exécution du programme, le canevas est affiché et le graphe est identique à celui que l'utilisateur a conçu.

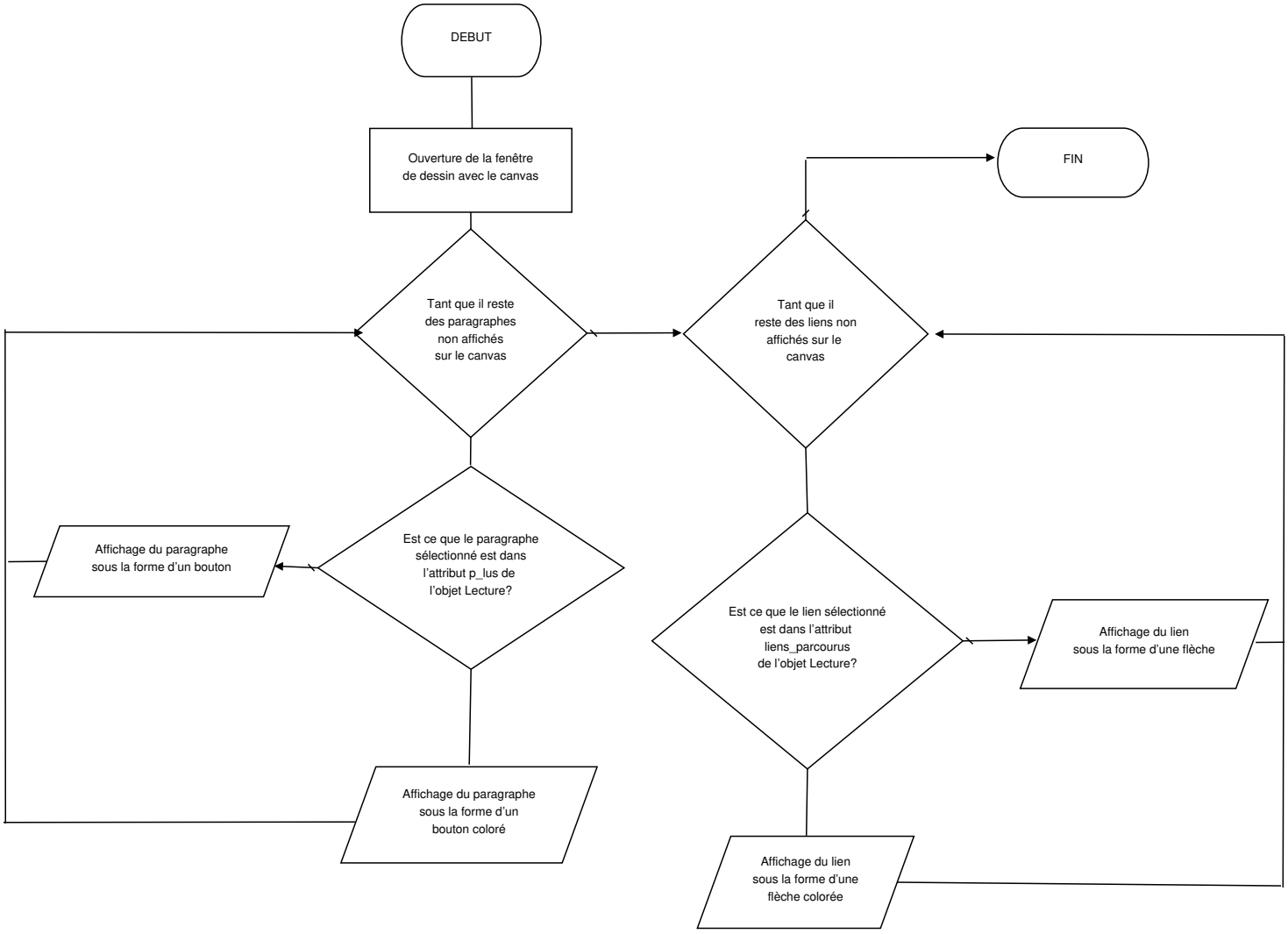


FIGURE III.5 – Algorithme de dessin du graphe

B Structures de données

Le choix des structures de données a été rapide. Nous avons dû choisir rapidement car une première question se posait dès la phase de conception des objets de notre application : Comment ranger nos objets ? Les paragraphes possèdent un ensemble de choix, un ensemble de liens, l'histoire est composée d'un ensemble de paragraphes, de liens et choix, la structure de donnée était au centre de la création des objets. Nous ne connaissions que deux objets Python permettant de gérer un ensemble : les listes et les dictionnaires.
Notre objectif était de pouvoir « ranger » nos objets, l'ordre n'étant pas important, de pouvoir en rajouter facilement, puis de pouvoir les retrouver si besoin en parcourant la structure de donnée.
Nous avons opté pour les listes pour tout les attributs et éléments qui nécessitait un rangement d'un ensemble d' objets. Nous nous sommes focalisés sur le fait que nous n'avions pas besoin d'un couple clé/valeur, amenant un moyen de « coder » chaque objet. La liste nous donne le moyen de gérer les objets grâce aux indices, mais nous n'avons pas usé de cette possibilité, ne voyant pas avec quelle fonctionnalité cela pouvait être utile.

C Bibliothèques utilisées

a Tkinter

Tkinter est une bibliothèque standard de Python. Celle-ci permet la création d'interfaces graphique adaptées pour des logiciels. Cette bibliothèque s'est avéré très utile pour former les différentes parties de l'éditeur de texte et du mode Lecture.
Nous avons utilisé pour notre fenêtre l'élément Canvas de Tkinter, ce qui nous a permis de placer les boutons représentant les paragraphes ainsi que de modéliser les liens reliant les paragraphes pour l'éditeur, et placer une image de fond pour le mode Lecture. De plus, afin de donner une apparence de livre au mode Lecture, nous avons utilisé l'élément PanedWindow pour séparer le texte et les choix du paragraphe en cours.
Les menus de création de lien et de choix, ainsi que de suppression utilisent les éléments ListBox, permettant de faire la liste de plusieurs objets et d'en sélectionner un, mais aussi la possibilité de Tkinter d'ajouter des événements. Nos menus peuvent ainsi se contrôler avec une pression de la touche entrée, pour éviter d'empiler les boutons de validation.

b Pickle

Pickle est une bibliothèque standard de Python permettant de sauvegarder des données python sous forme d'objets dans un fichier binaire. N'importe quel objet Python peut être sauvegardé dans ces fichiers, y compris des classes créées et définies dans d'autres fichiers Python. La problématique de la sauvegarde s'est posée assez tôt, pour permettre une transition vers le mode lecture. Néanmoins, conserver tout les paragraphes dans un fichier texte semblait difficile, pour plusieurs raisons :

- Le besoin de créer des conventions de sauvegarde du texte pour séparer les paragraphes (des sauts de ligne, un mot de fin)
- La difficulté de retrouver un texte particulier. Il aurait fallu inventer une sorte de code pour identifier chaque paragraphe, le tout inscrit dans le fichier texte
- La modification d'un texte aurait été impossible, le fichier texte ne permettant pas d'effacer une partie d'un texte pour le réécrire
- La sauvegarde des autres élèments essentiels au livre, comme les liens, qui ne sont pas des élèments textuels

Pickle offre un moyen de sauvegarder directement des objets, et de les récupérer de la même manière, ce qui enlève les difficultés présentées ci haut.
Pickle possède deux classes particulières : Pickler et Unpickler. L'objet Pickler est relié au fichier binaire ouvert, et permet d'écrire dans le fichier au moyen de la méthode dump. L'objet Unpickler est également relié au fichier binaire, mais permet de récupérer un objet sauvegardé avec la méthode load, qui va placer l'objet sauvegardé dans une variable du même type ou de la même classe.
Toutefois, il y a une contrainte à sauvegarder les objets Python instanciés dans d'autres fichiers Python grâce à Pickle. Pickle sauvegarde non seulement l'objet, mais aussi son chemin relatif, dont la racine est le fichier Python qui a instancié l'objet sauvegardé. Lors de la récupération de l'objet, l'objet est immédiatement associé à une variable de même classe. Cette variable doit avoir été instancié de la même manière que l'objet sauvegardé, et doit possèder le même chemin relatif. C'est une contrainte qui nous a obligé à modifier l'organisation de nos fichiers python, et qui oblige le fichier maître du mode lecture à se trouver au même niveau que le fichier maître de l'Éditeur de livre.
Nous avons choisi de sauvegarder uniquement l'objet Histoire dans un fichier, car celui ci contient déjà la référence à tout les objets crées par l'utilisateur. Ces derniers sont donc sauvegardés. Toute l'architecture du livre est conservée, ce qui permettrait, à terme, de recharger le livre dans l'Éditeur de livre pour permette d'éditer un livre.

Partie IV

Architecture du projet

A Diagramme de Classe

Nos objets étaient prévus pour être très liés entre eux dès le début, à l'exception de l'objet Lecteur, qui lui a été pensé pour le mode lecture. Néanmoins, la seule relation qu'ils ont en commun (sauf entre Choix et Lien) est le fait qu'ils se contiennent presque tous.

L'objet Histoire est réellement l'objet qui peut être assimilé au livre. Il contient l'ensemble des objets que l'utilisateur crée au sein de son histoire. Il possède des méthodes d'ajout pour chaque objet, ainsi que des méthodes de suppression pour chaque objet. Enfin, comme l'objet Histoire est celui qui est utilisé lors de la lecture, il possède une méthode pour le sauvegarder, ainsi que pour le charger dans le mode lecture.

Après l'objet Histoire, l'objet Paragraphe est le plus important. Il représente le texte du livre, divisé en petits bouts. En plus du contenu et de son titre, le paragraphe doit avoir un statut qui peut être :

- Fin
- Milieu
- Début

Ce statut sert à connaître la position du paragraphe dans le livre, si c'est un contenu au milieu de l'histoire, si il faut le lire au début, ou si au contraire, il annonce une fin. Les choix sont des morceaux de paragraphes spécifique, c'est donc évident qu'ils soient inclus dans un objet paragraphe. Puis, le paragraphe garde la trace des liens qui mènent vers lui, les liens Input symbolisés par liens_I, et des liens qui sortent, grâce à ses choix, les liens Output, symbolisés par liens_O. Enfin, sa position représente ses coordonnées dans la fenêtre.
L'objet Paragraphe possède beaucoup de méthodes. Comme méthodes importantes, nous pouvons citer *CreerPara* qui va compléter l'objet Paragraphe avec les renseignements de l'utilisateur, et *Affiche* qui permet d'afficher les renseignements du Paragraphe lors d'une action particulière (clic sur un bouton). L'objet Paragraphe possède des méthodes pour mettre à jour ses attributs, que ça soit pour les rajouter, les modifier (uniquement pour le contenu et le titre) ou alors pour supprimer des objets de ses attributs. Enfin, la méthode de suppression est assez complète car elle utilise également des fonctions de suppression des objets Choix et Liens, forcément affectés par cette suppression.

L'objet Choix est une partie du texte assez importante, qui va permettre d'orienter le lecteur vers un paragraphe particulier ou un autre. Outre son nom et son contenu, le Choix garde la trace de son « parent » qui est un paragraphe, et un attribut pour le lier à un lien.
L'objet Choix possède une méthode pour compléter ses attributs à partir des renseignements de l'utilisateur, une autre pour le lier à un lien et enfin une dernière pour effacer sa liaison avec un lien.

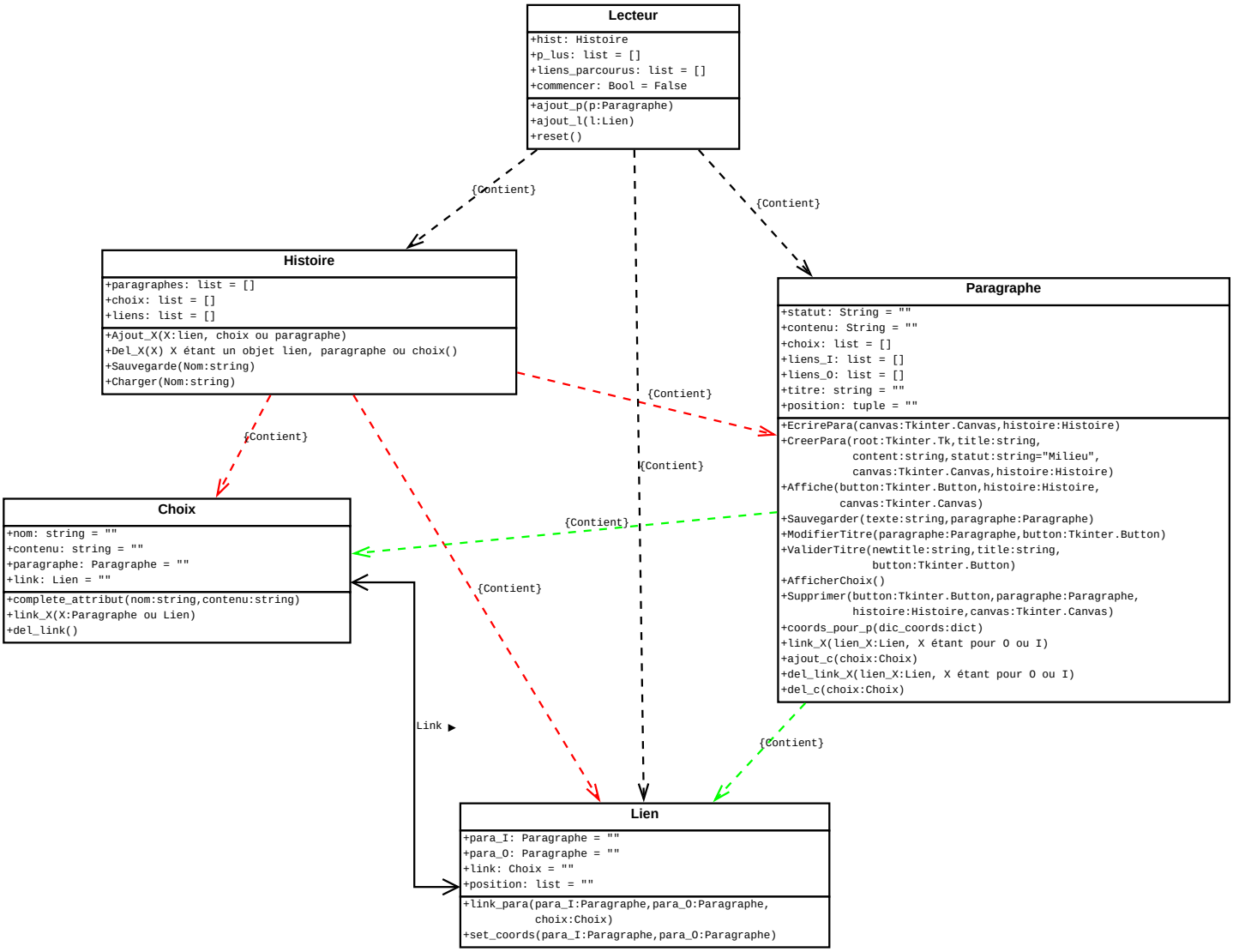


FIGURE IV.1 – Diagramme de classe de nos principaux objets

L'objet Lien est intangible, au niveau d'un livre, mais il est au centre des relations entre objets. Il garde la trace des deux paragraphes qu'il relie, et également au choix auquel il est lié. Cette liaison est importante car dans un livre dont vous êtes le héros, c'est le choix qui conduit à un paragraphe particulier. Dans notre reprśentation objets, le lien est l'intermédiaire entre le paragraphe, le choix et le second paragraphe. Enfin, le lien reprends les positions des deux paragraphes qu'il relie afin d'être reprśentable dans la fenêtre.
L'objet Lien possède uniquement deux méthodes, une pour le lier à tout les objets dont il a besoin pour exister, et une autre pour lui donner ses coordonnées.

L'objet Lecteur a une existence particulìere car il n'est instancié que pendant le Mode Lecture. Il sert à représenter le lecteur pendant le jeu, ce qui permet de suivre sa progression dans le livre, d'où l'existence de p_lus, qui reprèsente les paragraphes lus, et liens_parcourus, qui reprèsente les « chemins » empruntés par le joueur. L'objet possède également l'histoire que le joueur est en train de lire (logique, si l'objet Lecteur est la représentation du joueur). Enfin, l'objet contient un dernier attribut, un attribut booléen qui permet de savoir si le joueur a commencé le jeu ou non.

Les méthodes de l'objet Lecteur se limitent à ajouter des objets à ses différents attributs, mais aussi à réinitialiser ces derniers. En effet, si le joueur recommence, son ancienne progression doit être oublié pour pouvoir enregistrer celle de sa nouvelle partie.
La partie graphique n'a pas été vraiment pensée lors de la conception, aussi, il n'est pas intégré dans le diagramme des classes. Néanmoins, l'interface graphique est pensée en terme de classes, et le menu recense presque toutes les fonctionnalités implantées dans notre logiciel.

B Exemples d'utilisations

a Utilisateur de l'Éditeur de Livre

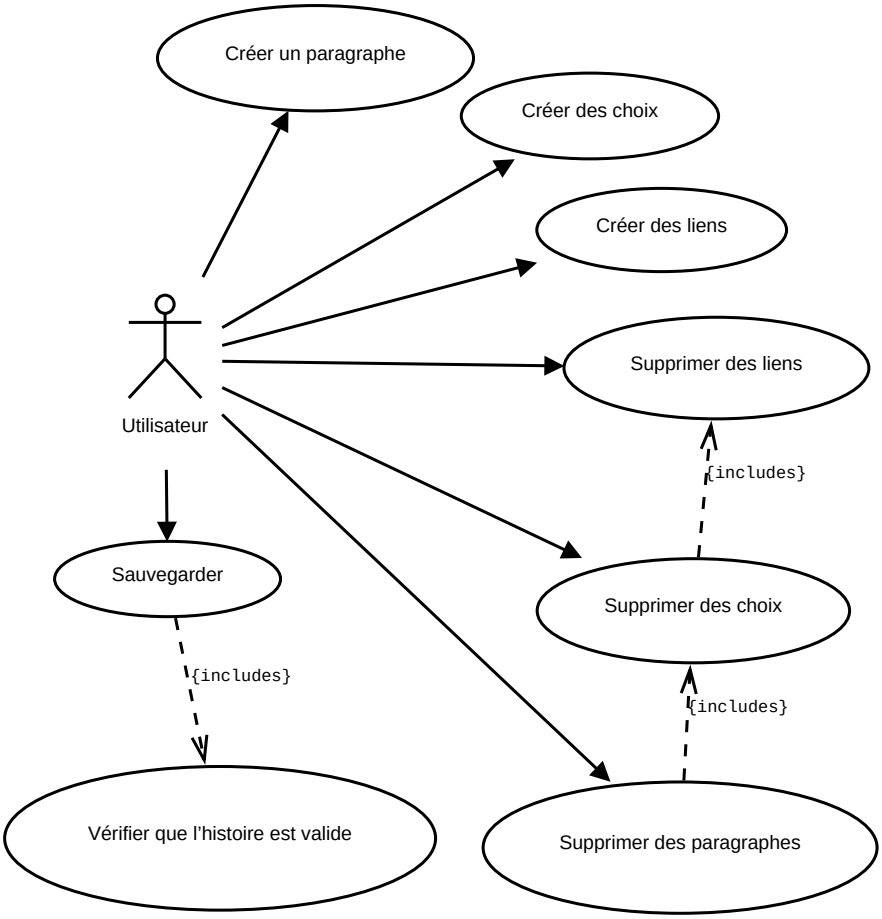


FIGURE IV.2 – Diagramme d'utilisateur de l'Éditeur de Livre

L'utilisateur de l'Éditeur de Livre a accès à toutes les fonctionnalités lui permettant de créer son livre et de le modifier. Toutes les possibilités de création sont disponibles, ainsi que celles de suppression. Les fonctionnalités de suppressions s'incluent récursivement les unes dans les autres, afin que l'utilisateur n'ai pas besoin de s'assurer que tout les élements d'un objet ont été supprimé afin de pouvoir le supprimer. Enfin, la possibilité de sauvegarder son livre inclut la vérification automatique de la validité du livre (présence d'une fin gagnante et possibilité d'emprunter tout les chemins possibles), fonctionnalité qui peut être utilisée sans sauvegarder le livre, mais qui a beaucoup plus de sens associé à la sauvegarde.

b Utilisateur du Mode Lecture

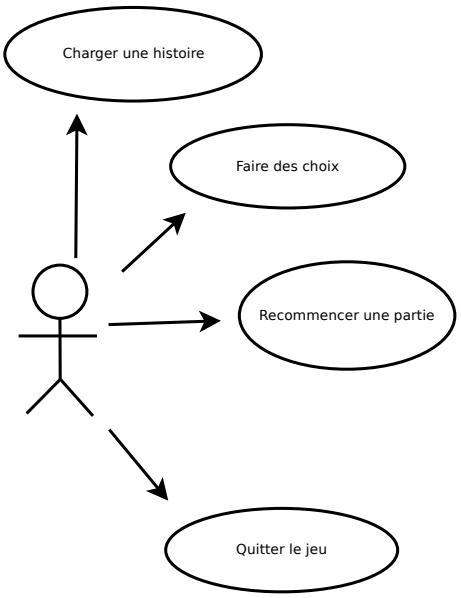


FIGURE IV.3 – Diagramme d'utilisateur du Mode Lecture

Lorsqu'il lance le Mode Lecture, l'utilisateur est invité à choisir entre deux options. Soit quitter le Mode Lecture, soit continuer afin de charger son histoire. Une fois cela fait, l'utilisateur peut jouer. Le premier paragraphe se charge, ainsi que ses choix. Le joueur peut, de ce fait, cliquer sur les choix proposés, et un nouveau paragraphe se charge jusqu'à ce que l'on arrive à la fin. Une fois le dernier bouton cliqué (que ce soit gagnant ou non), une image du graphe tel qu'on le voit en mode Éditeur de texte s'affiche avec une coloration pour le chemin par lequel l'utilisateur est passé, Une fois cela exploré, une derniÃ¨re fenêtre s'affiche pour demander si l'on souhaite rejouer ou quitter le jeu.

Partie V

Expérimentation et usages

A Visuel de l'application

a Graphe

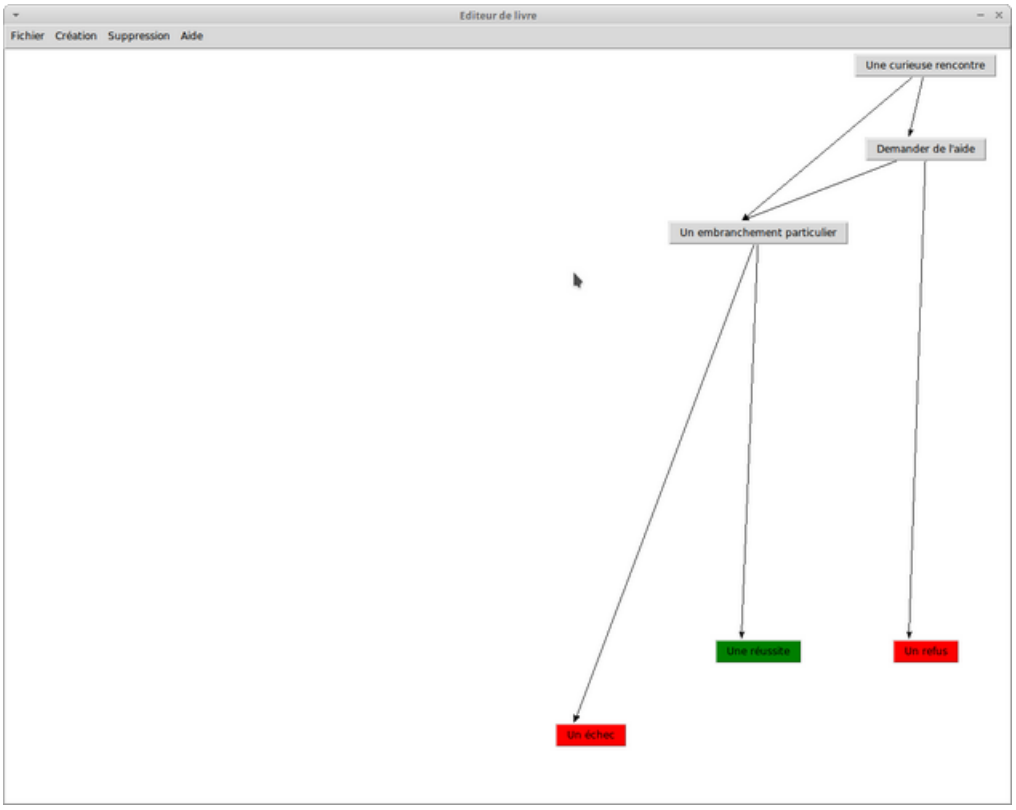


FIGURE V.1 – Aperçu du graphe

Le graphe est décomposé en 9 lignes et 6 colonnes. La première ligne est exclusivement composé de paragraphe de début, les 6 suivantes comportent des paragraphes de milieu et les deux derniÃ¨res lignes sont réservées pour les paragraphes de fin. Les paragraphes de fin sont différenciés en deux couleurs : vert pour les fins gagnantes et rouge pour les perdantes.
Lorsque l'on clique sur un paragraphe, une fenêtre s'ouvre afin d'afficher les options de modification et de suppression. Enfin, lors de la crátion d'un lien, celui-ci s'affiche sous la forme d'une flèche qui relie les deux paragraphes.

b Visuel d'un menu de création

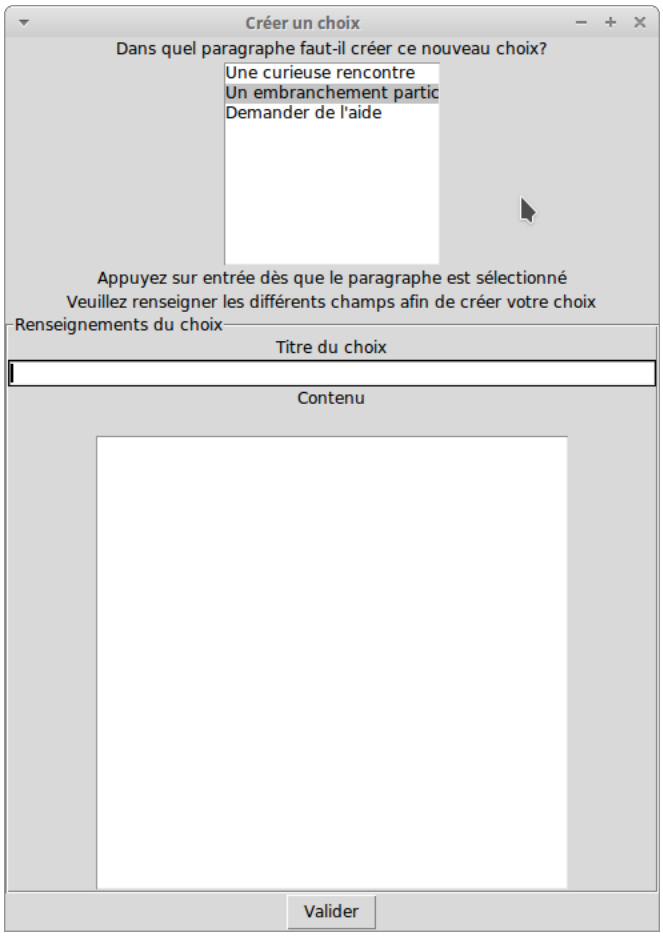


FIGURE V.2 – Menu de création d'un choix

Les menus de création sont sensiblement identiques, mis à part pour la création des liens, qui comporte uniquement des éléments ListBox.
La fenêtre de création s'ouvre lorsque l'utilisateur appuie sur une commande du menu création. Mais la fenêtre n'est pas encore complète. En effet, pour capter au fur et à mesure les informations, nous avons décidé de faire apparaître les différents points de la fenêtre au fur et à mesure que l'utilisateur appuie sur la touche entrée.
Dans l'exemple donné Figure V.2, seul la ListBox s'affiche lors de l'ouverture de la fenêtre. Lorsque l'utilisateur sélectionne un élément et appuie sur entrée, la partie « Renseignements du choix » apparait. Le bouton « Valider » apparait également, pour éviter à l'utilisateur de valider la création de son choix alors qu'il faisait un saut de ligne avec la touche entrée.
Les menus de suppression sont quasiment identiques à ceux de création (pour les choix et les liens), mis à part qu'il n'y a aucun champ de texte présent. En effet, pour supprimer un élément, il faut juste renseigner lequel, et les ListBox sont adaptées à cette utilisation.

B Performances

Malheureusement à cause du manque de temps effectif, nous n'avons pas pu effectuer de vrais tests de performances. Néanmoins nous pouvons tirer des conséquences de nos choix de développement à partir des connaissances que nous avons acquises tout le long de ce projet.

a Faiblesse des structures de données

À première vue, notre choix de structure de données nous paraissait bon, et il fonctionnait bien avec les fonctionnalités que nous voulions développer.
Néanmoins, la majorité de nos structures de données n'utilisent pas la propriété d'ordre des listes : nous ne cherchons jamais un objet à partir de son indice. Dans cette optique, l'objet set fourni nativement par Python semble beaucoup plus adapté à nos applications.
Le set est une collection d'objet non ordonnés et tous uniques, ce qui correspond à une collection de paragraphes, de lien ou encore de choix, l'ordre n'a aucune importance. La vitesse d'itération est sensiblement la même, ainsi que la vitesse de création à vide. En revanche, la vitesse de suppression d'un élement dans un set est constante, alors que dans une liste, elle dépend de sa longueur. Nos structures de données sont susceptibles de contenir beaucoup d'élements : si nous créons 50 paragraphes (le maximum pour notre application) et 2 choix pour chaque paragraphe, nous obtenons des listes avec 50 ou 100 élèments, ce qui rends la suppression beaucoup plus lente au fur et à mesure que nous augmentons la longueur des listes.
Il y a donc une baisse de performance de l'application liée à ses structures de données, mais cela reste assez peu visible car cela n'affecte que la capacité de suppression des objets, et seulement sur de grandes quantités d'objets.

b Instanciation des objets

Le second point qui semble être un point faible dans notre application est l'instanciation de nos différents objets. La création de nos objets suivent le même schéma, à savoir :

- L'objet est créé, donc instancié
- Une méthode est presque directement utilisée afin de compléter ses attributs essentiels à sa création effective.

Outre le fait que certains attributs ne sont pas nécessairement complété lors de cette phase (un paragraphe n'a pas de choix, ni de liens à sa création, ce sont des listes vides), il y a quand même une grande perte de temps à effectuer l'instanciation en deux fois.
Pour gagner du temps lors de la création des objets, nous pourrions mettre des arguments obligatoires lors de la création d'un objet d'une classe donnée. Par exemple, un lien aura forcément tout ses attributs non vides après sa création, c'est un objet immuable. Au lieu d'avoir une méthode pour compléter ses attributs et une fonction pour génerer un objet Lien, nous pourrions tout simplement mettre en argument tout les renseignements nécessaires à sa création, et économiser du temps d'exécution, ainsi que des lignes de codes dans nos fichier.

C Bugs rencontrés

Nous allons parler rapidement des bugs les plus gênants rencontrés durant l'élaboration de notre logiciel. Il nous semble important d'en parler car ils se sont révélés au fur et à mesure de nos expérimentations, malheureusement assez tardives, ce qui ne nous a pas permis de tous les corriger.

a Deux liens pour un choix

Après notre mise en commun, au moment où l'Éditeur de livre commençait à devenir fonctionnel, nous nous sommes aperçu que si un choix pouvait bel est bien être lié à un lien, il restait possible de le sélectionner dans le menu de création de liens, et ainsi lui attribuer un autre lien.
Dans l'attribut lien de choix, le nouveau lien écrasait l'ancien, mais la flèche restait sur le canevas, prêtant à confusion.

Pour éviter ce cas de figure, nous avons préféré empêcher l'affichage du choix si il était déjà lié à un lien. A l'avenir, une solution plus poussée serait de détruire le lien déjà existant pour le remplacer par le nouveau.

b Limite de paragraphe

Notre logiciel est prévu pour accueillir 54 paragraphes selon la répartition suivante :

- 6 paragraphes de statut « Début »
- 36 paragraphes de statut « Milieu »
- 12 paragraphes de statut « Fin »

Or, nous n'avions rien prévu si l'utilisateur cherchait à créer un paragraphe alors que la limite était atteinte. Il en résultait des bugs en cascade.
Le bug se produisait au moment où le paragraphe cherchait des coordonnées, grâce à un dictionnaire recensant toutes les positions possibles pour un paragraphe sur le canevas. La clé est le tuple de coordonnées, la valeur est un booléen, pour indiquer si les coordonnées sont prises ou non. Il y a un parcours du dictionnaire jusqu'à trouver des coordonnées libres pour un paragraphe avec un statut donné. Si jamais aucun coordonnée n'est possible, la fonction ne retournait rien, et le bouton associé au paragraphe essayait de se placer sur le canevas, sans coordonnées, ce qui produisait une erreur.
Pour palier à ce problème, nous avons instauré un code d'erreur que retourne la fonction chargée de chercher des coordonnées au paragraphe. Si le code d'erreur est captée, la création du paragraphe est interrompue et un message d'erreur s'affiche.

c Suppression d'un paragraphe en bout de lien

Lors de la suppression d'un paragraphe, les objets qu'ils contient sont supprimés, notamment les choix, et la suppression des choix entraine la suppression des liens éventuels liés à ces choix. Supprimer un paragraphe revient donc à supprimer tout les liens démarrant de ce paragraphe.
Nous avons néanmoins remarqué que, au vu de l'algorithme adopté pour la suppression d'un paragraphe, il n'y a avait rien pour supprimer les liens qui menaient vers ce paragraphe. Il en résulte un « trou » dans l'histoire, une flèche ne menant à rien sur le canevas, ainsi qu'une ou plusieurs référence au paragraphe censé être effacé. Si l'utilisateur ne pense pas à supprimer ce lien désormais inutile, le livre ne pourra pas être lu par le mode lecture, le paragraphe n'existant plus.
Nous avons simplement modifié nos algorithmes de suppression, notamment celui de suppression des liens, qui prends désormais en compte la position du paragraphe dans le lien, et nous exécutons cet algorithme dans l'algorithme de suppression de paragraphe.

d Un lien qui est de type str

Quatre heures avant la fin de notre projet, lorsque nous faisions communiquer les fichiers créés avec notre Éditeur de Livre vers notre Mode Lecture, nous découvrons un bug très gênant qui empêche la lecture du livre. Lorsque le joueur clique sur un choix, il est possible qu'une erreur python s'affiche, disant qu'un objet de type str n'a pas d'attribut permettant de trouver le paragraphe suivant, ce qui bloque tout le jeu.
Le lien qui est retrouvé est étrangement de type str, au lieu d'être de type Lien. Nous cherchons l'origine dans la création du lien, car la sauvegarde n'est pas en cause. Après de multiples essais, de vérifications avec des affichage à chaque lien créé, nous découvrons que si deux choix ont exactement le même attribut nom, il y a un problème dans la création du lien menant au deuxième choix.
Ce bug provient de la manière dont nous récupérons le choix, utilisant son attribut nom afin de permettre à l'utilisateur de mieux le reconnaître dans une liste affichée à l'écran. Une fois reconnu, ce bug a été très simple à régler. Il a suffit de restreindre la recherche d'un choix Ã la liste de choix du paragraphe dont il est issu. En effet, lorsque nous affichons les choix, nous affichons les choix d'un seul paragraphe. Il est donc logique de retrouver le choix uniquement dans la liste des choix de ce même paragraphe. De plus, nous améliorons nos performances en itérant sur une liste plus petite !

e La suppression du lien sur le canevas

Ce bug a été découvert par hasard, et n'est peut être pas réglé pour tout les cas où il pourrait se présenter.
Lors de la suppression d'un lien, un paragraphe pouvait disparaitre du canevas, et la flèche du lien restait. Mais l'objet Paragraphe en lui même existe toujours et il est possible de créer d'autres liens, des choix et de sauvegarder son histoire.
Ce bug est provoqué par notre manière de supprimer la flèche. Pour supprimer un élément du canevas, il faut l'identifier par un tag. Nous associons la flèche avec un tag, puis nous supprimons la flèche en passant ce même tag en argument. Nous attribuons un tag à la flèche grâce à des coordonnées. Sauf que cette méthode cible l'élément le plus proche de ces coordonnées. Si le bouton d'un paragraphe est plus près que le lien, c'est lui qui sera ciblé et donc effacé.
Pour corriger ce bug, nous avons essayé de ne pas donner exactement une partie des coordonnées de la flèche, mais de donner le point du milieu de la flèche, pour être sûr que c'est la flèche qui recevra le tag. Mais nous ne sommes pas sûr que cela marche pour toutes les situations.

Partie VI

Conclusion

A Points clés du projet

Lors de la création ou la suppression d'éléments dans le graphe, que ce soit un paragraphe, un lien ou un choix, tout passe par la classe Histoire qui centralise chaque élément ajouté et qui symbolise le livre, ce qui facilite la lecture.
Les paragraphes étant le coeur du graphe (ils contiennent les choix et les liens liés à eux), il fallait que chaque attribut de la classe soit référencé clairement et de façon détaillée afin de faciliter leur manipulation dans d'autre fichier. C'est pour cela que lors de leur création, il est demandé d'entrer des paramètres tel qu'un titre et son statut dans le graphe.
Enfin, notre capacité d'enregistrer directement l'objet de classe Histoire grâce à la bibliothèque Pickle nous permet de simplifier grandement son chargement dans le mode lecture.

B Améliorations techniques envisageables

Pour conclure notre rapport, nous allons vous présenter quelques améliorations importantes et techniques que nous avions envisagé pour notre projet.

a Changement de structures de données

Le changement de structure de données est un travail énorme, mais qui pourrait devenir payant en terme de performances. Si cela était possible, nous voudrions changer la plupart de nos structures de données de type list en type set, comme expliqué Partie V Section B sous-section a. Comme le type list et set sont très proches, nos lignes de code changeront peu, l'itération est identique et presque toutes les méthodes le sont. Cependant les performances de notre applications seront améliorées.

b Ajout d'images dans les paragraphes lors de la création pour enrichir le mode lecture

Notre première idée pour le mode lecture était de créer des thèmes de couleur que l'utilisateur pourrait sélectionner lors de la création de ses paragraphes. Ensuite dans le mode lecture, les paragraphes apparaitraient avec un fond de couleurs différentes, selon leurs thèmes. Cette idée ayant été refusé, nous avons procédé autrement, en perdant cette personnalisation.
Néanmoins, nous pourrions proposer à l'utilisateur d'intégrer ses propres images pour chaque paragraphe, qui viendront agrémenter le mode lecture. Le joueur pourra ainsi voir son jeu accompagné d'une illustration représentant ce qu'il lit à l'écran, comme dans un vrai livre. Il suffirait pour cela d'associer le nom d'un fichier dans un dossier particulier à un paragraphe, pour ensuite le faire afficher si il existe.

c Affichage des liens présents dans le menu des paragraphes

Lors de la création de lien, ceux-ci ne font que s'afficher dans le canevas. Afin d'améliorer la visibilité des liens, nous pourrions mettre les liens en surbrillance lorsque l'on survol un paragraphe avec la souris ou encore les afficher dans la fenêtre de paragraphe.