Cahier des charges CodeLab 3

I - Description du projet

Le but de ce projet est de concevoir un outil pédagogique simple d'utilisation permettant d'enseigner la programmation aux débutants. Le produit doit prendre la forme d'un laboratoire de programmation avec une interface graphique codée en Java. Le laboratoire doit présenter plusieurs exercices de même nature à l'utilisateur qui devra les compléter en développant des programmes. Il faudra donc fournir un langage de programmation et une interface d'édition de code. Ensuite, il faut que l'utilisateur puisse observer les effets de son code à travers l'interface. Il faudra donc créer des exercices ayant une représentation visuelle claire, et permettre de tester un programme sur un exercice.

II - Objectif

Nous voulons à travers nos niveaux pouvoir permettre aux néophytes de la programmation de comprendre les concepts d'algorithmes, de boucle, de conditions, en bref de programmation de façon amusante.

Pour pouvoir leur introduire ces concepts de la programmation, nous avons comme objectif de mettre en place une histoire que le joueur devra suivre (donc pas de possibilité de choisir arbitrairement un niveau non fait). L'histoire devra introduire pas à pas ces concepts par des mécaniques de gameplay et par un Level design au joueur. Comme tout programme ne se valent pas, nous voudrions ajouter un système de score (3 étoiles) pour pouvoir récompenser les joueurs qui proposent du code efficace.

En plus de tout cela, nous voudrions ajouter optionnellement des mécaniques de gameplay additionnelle comme des objets à collecter,... (cf Partie Option pour plus de détails sur ces mécaniques) pour ajouter du challenge aux différents niveaux.

Si possible, il faudrait laisser la possibilité aux joueurs de pouvoir créer ses propres niveaux en utilisant les différentes mécaniques de gameplay/level design et de pouvoir les essayer et les sauvegarder dans le laboratoire.

III - Notre implémentation

1) Produit minimal viable :

Notre produit minimal doit être composé d'un seul niveau contenant une zone de terrain avec un personnage sur une case du terrain. Il faut aussi une zone pour pouvoir coder des instructions simples (Avancer, tourner à 90 degrés, etc) et les exécuter.
Lors de l'exécution, il faudra que le personnage soit capable de bouger et que tout type d'instruction présent dans l'intégralité du CodeLab fonctionne.
A la fin du niveau, l'algorithme doit savoir si le joueur est bien arrivé au point d'arrivé.

2) Les Objectifs du joueur :

L'objectif du joueur est d'aller d'un point A vers un point B en utilisant un programme dédié.

Sur le terrain il y aurait potentiellement des collectables à ramasser pour pouvoir faire gonfler le score du joueur (pièces) et lui permettre d'atteindre les trois étoiles (un score présent noté en étoile s'il est implémenté cf. options) ou bien de tout simplement terminer le niveau (clef). Le programme à créer doit être le plus efficace possible pour pouvoir augmenter le score. Les actions possibles par le joueur seront débloquées (exemple: impossibilité d'utiliser des "if" avant un certain niveau) en fonction des niveaux ou simplement d'interdire certaines fonctions selon le niveau (sûrement vers les derniers niveaux les plus ardus).

3) Language

Nous estimons que l'édition de code en blocs que l'utilisateur manipule avec des glisser-déposer (drag and drop) est plus pédagogique. Nous allons donc implémenter l'édition de code avec ce système utilisant trois types de blocs :

-Les premiers blocs seront les instructions élémentaires. Ce sont les actions de déplacement (et potentiellement les actions de combat). Chaque instruction élémentaire incrémentera un compteur d'action qui nous permettra d'imposer des contraintes pour les exercices.

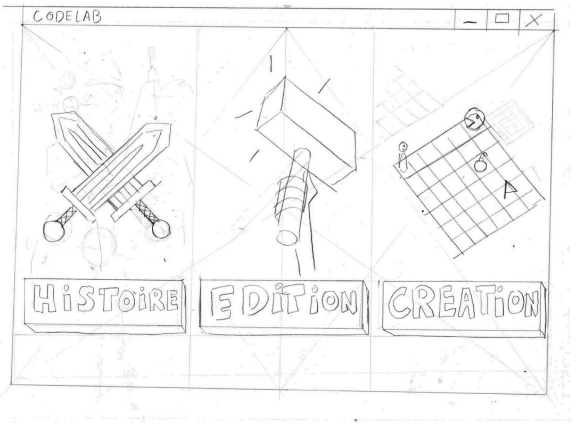
-Les seconds blocs seront les structures de contrôle. Nous allons implémenter les conditions "if", ainsi que des boucles "while" et "for". Les appels à ces blocs n'incrémenteront pas le compteur d'actions élémentaires.

-Les derniers blocs seront donc les conditions elles-mêmes. Lorsque le personnage a atteint l'arrivée, lorsqu'un obstacle se trouve devant, lorsqu'une bifurcation se présente… Ces blocs ne gonfleront pas non plus le compteur d'action.

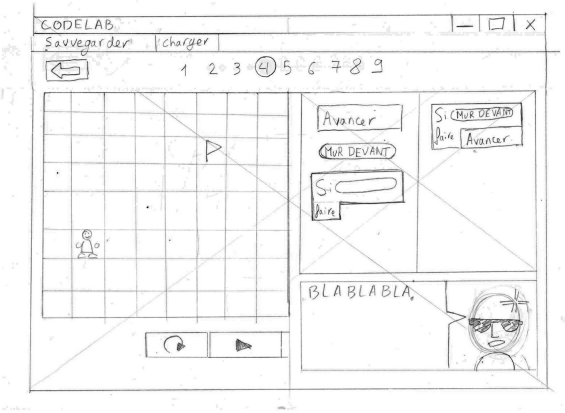
- Nous avons pensé à ces éléments de langage pour l'instant : "avancer", "tourner"(gauche, droite à 90 degrés), "attendre", "tant que", "si", condition (pic devant , vide, mur…). Le but est d'avoir un pseudo code pour une meilleure compréhension pédagogique).

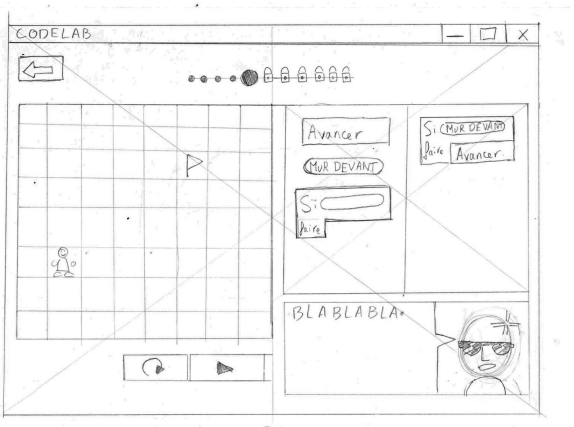
4) Interface Graphique

- 2 idées d'écran d'accueil : (Il doit contenir 3 boutons : une pour l'histoire, une pour accéder à l'édition de niveau et une pour charger les créations)

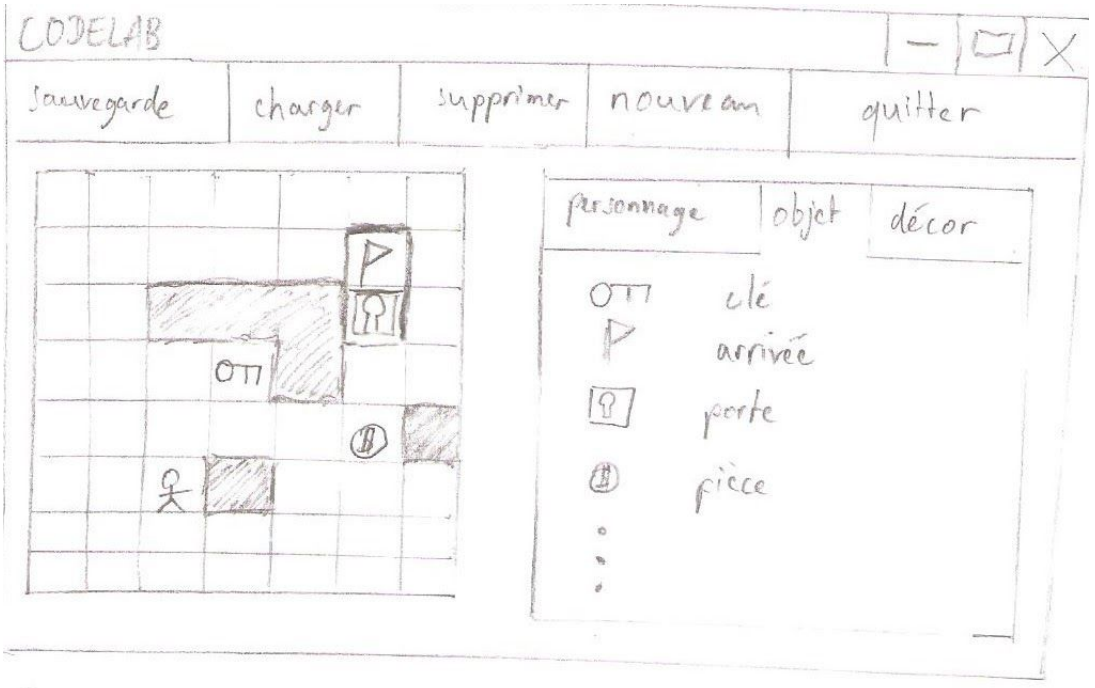


Fenêtre de l'histoire avec option sauvegarde + charge ou sans :

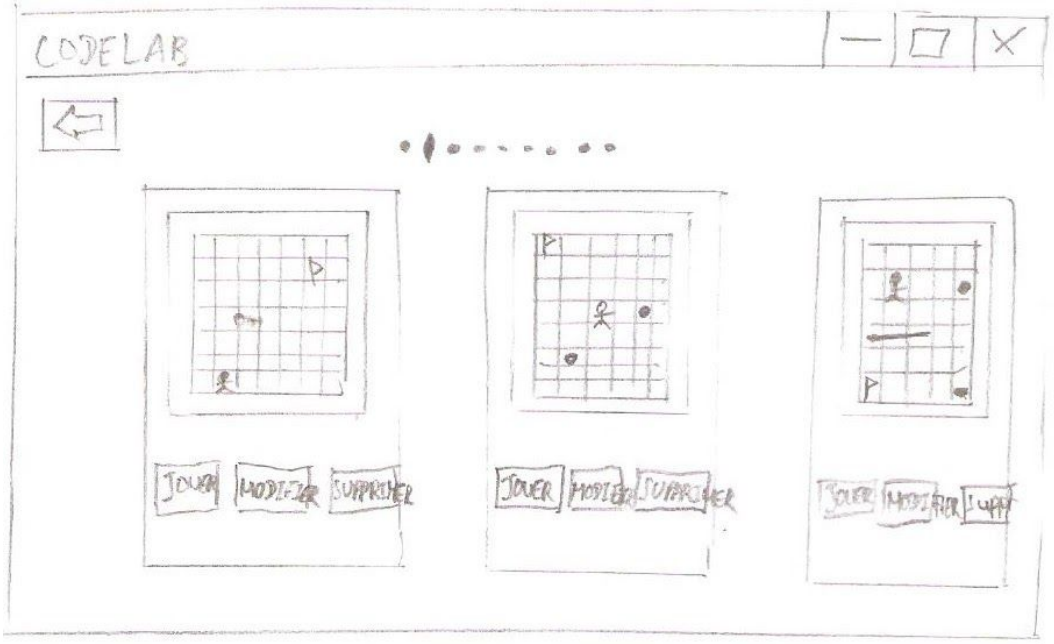




Editeur de niveau :



Création : (Charger les niveaux créer par les joueurs)



IV Options et ajout

1) Editeur de Niveau

Le laboratoire doit également être fourni avec des outils de gestion de fichiers correspondant aux exercices. Il doit être apte à créer des exercices avec un éditeur, de les sauvegarder, les modifier, et de les supprimer. L'éditeur devra être capable de déterminer si le code entré par l'utilisateur est syntaxiquement correct. De plus, chaque exercice doit être équipé d'un programme à même de déterminer si le code de l'utilisateur permet de résoudre le problème de l'exercice (autre possibilité complémentaire serait de forcer le créateur à terminer lui-même son niveau afin de le valider).

2) Histoire

Voici plusieurs suggestion d'histoire pour le jeu pour rendre le rendre plus immersif et amusant qu'un simple "laboratoire de code" :

- Vous êtes un duo ayant pour but de prendre la tête du Roi et de semer la panique dans le Royaume mais pour pouvoir atteindre ce Roi il va falloir traverser plusieurs étages en bravant moults dangers. Vous devrez guidez votre ami dans les salles pour déjouer leurs pièges et provoquer la révolution.
- King Kong et son frère Donkey ont volé votre banane et se sont cachés dans leur forêt remplie de pièges ! Quelle bande de méchants ! Allez reprendre votre goûter en toute discrétion car franchement les bananes c'est la vie !
- Vous êtes un professeur archéologue londonien gentleman vêtu d'un haut de forme très distinctif, lors d'une exploration dans un temple antique, vous êtes séparé de votre apprenti. Dépêchez-vous de le retrouver pour percer les mystères de ces vestiges recelant le destin perdu d'un peuple !
- Vous êtes le 9ème participant d'une expérience scientifique, respectivement dans 9 bâtiments vous avez 9 heures pour en sortir et rejoindre les autres sujets. En revanche vous devrez passer à travers K portes pour sortir de ce 9ème bâtiment. Au total K devra être congru à 9. Attention l'équipe Nonary composée de 9 chercheurs vous interdit d'utiliser du Potassium !

-Croyez-vous au complot ? [ ] Oui
[ ] Non
[x] Chht !! Un agent de la CIA nous observe..

Cet agent c'est vous, explorez des régions et infiltrez des groupes pour démanteler des réseaux frauduleux ou protéger les concitoyens de menaces dignes d'un 007. Les complots sont faciles il va donc falloir sachoir éviter les scandales.

3) Mécaniques de Jeu

Voici nos idées pour rendre notre jeu plus original et plus complexe:

- Meilleur score possible (Le moins de “temps” possible) : Système de récompense pour les joueurs qui auront pris le moins d’actions élémentaire pour terminer un niveau (chaque niveau serait probablement noté sur 3 étoiles, le but étant de terminer l’épreuve, décrocher ce score serait une option).

- Collectable : Des objets seront disposés sur le chemin pour récompenser les joueurs qui prendront le risque pour aller les chercher (exemple: récupérer tel objet pourrait rajouter une étoile à votre score).

- Ennemis + paterne : Nous avons imaginé un gameplay, lorsque le joueur bouge sur le terrain, l’ennemi fait de même. Les ennemis auront un paterne différent qui seront simple à comprendre (exemple: cavalier qui avance de 2 cases devant en suivant un paterne défini à l’avance). Si l’ennemi touche le joueur alors le niveau est terminé et il devra recommencer ce niveau.

- Sol instable, Mur, vide, plateforme mobile : On aimerait également intégrer une diversification de l’environnement permettant de varier les situations poussant le joueur à être créatif et/ou efficace. (exemple: un ennemi qui vole pourra traverser le vide mais pas le joueur)

- Tour par tour : Donner la possibilité au joueur de taper plusieurs fois son code pour un niveau, par exemple il peut produire un programme pour atteindre une clef puis stopper le programme pour mettre un autre programme plus adapté pour atteindre la sortie. On pourrait limiter le nombre de tour pour éviter qu’il n’abuse de cette fonctionnalité (A voir avec le product owner).