Cahier des charges de CodeLab4 - *L'aventure*

Benjamin Nicolas
Hugo Thomas
Emile Rolley
Antoine Liu

4 février 2020

Résumé

Ce document présente ce que devra être `CodeLab4 - L'aventure` (CL4) et comment il devra être utilisé.

Table des matières

1 Présentation

CL4 est un logiciel ayant comme objectif de s'initier à la programmation de façons ludique et ergonomique, en effet, les problèmes à résoudre sont camouflés par un scénario permettant l'immersion de l'utilisateur.

L'utilisateur va devoir aider un naufragé à récupérer des ressources pour pouvoir construire un radeau et retrouver la civilisation. Mais attention ! Il ne devra pas mourir de faim ni se faire surprendre par une tempête.

CL4 est construit autour d'un langage simplifié permettant de se familiariser avec les conditions ainsi que les boucles. L'utilisateur doit écrire un ensemble d'instructions permettant au fur et à mesure de l'aventure de remplir des quêtes. Une partie est divisée en plusieurs chapitres de plus en plus coriaces à résoudre.

1.1 Objectifs

La version final de CL4 devra contenir :
— Un `launcher` permettant de créer et charger des sessions.
— Un module permettant de ouvrir et sauvegarder des programmes.
— Un éditeur permettant de modifier un programme.
— Un module permettant la visualisation de l'exécution d'un programme.
— Un module permettant de sauvegarder la progression de chaque session.
— Une GUI regroupant tous ces modules.

2 Utilisation

Une fois le `launcher` lancé, l'utilisateur doit choisir une session si il y en a au moins déjà une, ou bien il peut choisir de lancer une nouvelle partie en créant une nouvelle session.

La fenêtre principale se lance après que la session soit chargée. L'utilisateur peut alors résoudre les quêtes proposées grâce à l'éditeur de programme.

3 Déroulement d'un chapitre

3.1 Aspect général d'un chapitre

Un chapitre est constitué d'une ou de plusieurs quêtes à remplir. Une quête consiste en l'accomplissement d'une tâche simple ayant pour but d'emmener le joueur au prochain chapitre ou à la victoire.

Par exemple, un chapitre peut mener le joueur vers la construction d'un outil. Ce même chapitre sera divisé en quêtes, les premières quêtes pourrait être la récupération des ressources nécessaire à sa confection et puis la dernière quête pourrait être la création de cet outil.

Il faut donc pour chaque quête, fournir une solution (un ensemble d'instruction) répondant aux problèmes. La solution est considérée comme étant valide si et seulement si à la fin de l'exécution des instructions l'objectif de la quête en cours est complété. Par exemple, si l'intitulé de la quête est : "Trouver 3 morceaux de bois." ; il faut qu'à la fin l'inventaire du personnage contienne au moins 3 morceaux de bois.

3.2 Évolution des contraintes

Au fil des quêtes, le joueur disposera d'une jauge de "faim" qui diminuera à chaque actions effectuées. Il devra donc passer par les cases contenant de la nourriture afin de remplir sa barre de faim. D'autre part, la pluie pourra apparaître et le joueur ne devra pas se trouver sous celle-ci. (elle sera indiquée sous la forme "n heures de pluie dans m tours"). Le joueur sera donc forcé

d'utiliser des boucles, qui diminueront le temps d'une instruction (au lieu de faire perdre deux unités de faim/temps elle en fera perdre qu'une seule) pour pouvoir effectuer une quête à temps.

4 Description de la GUI

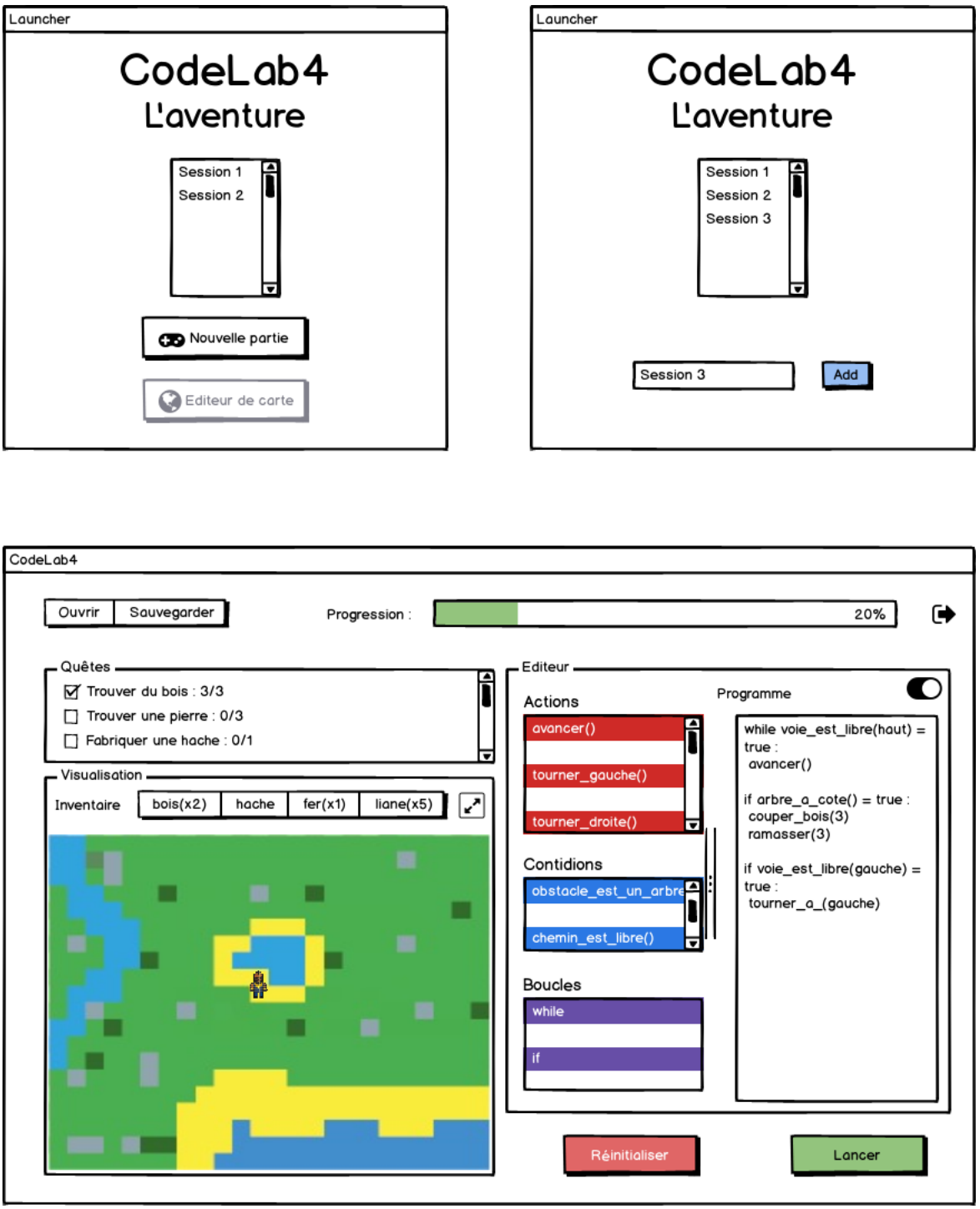

FIGURE 1 – Schéma de la GUI et du `launcher`

4.1 Le `launcher`

La liste des sessions correspond aux parties enregistrées par les utilisateurs.

Le bouton "nouvelle partie" lance une nouvelle session.

Le bouton "éditeur de carte" permet à un utilisateur de créer sa propre carte afin de partir à l'aventures sur sa création !

4.2 La fenêtre principale

L'affichage est divisé en deux panneaux principaux et d'autres affichages complémentaires.

4.2.1 Fenêtre de visualisation

La fenêtre de visualisation affiche l'état actuel de la carte et de l'inventaire. La carte est modifiée au cours de la partie selon les actions effectuées.(coupe des arbres, casse des cailloux, etc...) Elle est donc unique à chaque partie. L'inventaire correspond à ce que possède le joueur. Il lui permet donc de stocker des ressources afin de construire des outils et progresser dans l'aventure.

4.2.2 Éditeur de programme

L'éditeur est lui aussi séparé en deux parties. La partie "programme" permet à l'utilisateur de pouvoir écrire les suites d'instructions nécessaires à la résolution des quêtes. La partie de gauche contient les possibilités que l'utilisateur a à disposition, séparée en 3 catégories pour plus de clarté : les **actions**, les **conditions**, et les **mots clés**.

4.2.3 Parties complémentaires

Elles regroupent :
- Le panneau affichant les quêtes du chapitre actuel.
- La barre de progression indiquant à l'utilisateur son évolution par rapport à l'histoire.
- Le bouton "Jouer" lance l'exécution du programme de l'utilisateur.
- Le bouton "Réinitialiser" arrête l'exécution du programme et réinitialise l'état de la carte et de l'inventaire.
- Le bouton "Sauvegarder" permet de sauvegarder le contenu d'un programme dans un fichier texte.
- Le bouton "Ouvrir" permet d'ouvrir un fichier texte contenant du code préalablement enregistré.

5 Description du langage

5.1 Les actions

Au cours d'une partie, l'utilisateur doit donner une liste d'actions à effectuer (selon ou non certaines conditions), que le personnage devra accomplir afin d'atteindre divers objectifs, cette liste d'instructions forme un programme. Chaque ligne d'un programme ne peut contenir qu'une seule instruction.

Les actions sont représentées sous forme d'appel de fonctions et sont séparées en deux types :
- Les actions de mouvement : `tourner_a_gauche()`, `avancer()`, etc...
- Les actions d'interaction : `couper_bois()`, `manger()`, etc...

Celles-ci auront bien évidemment un coût (nourriture, temps), c'est la contrainte majeure qui obligera l'utilisateur à gérer ses actions.

5.2 Boucles et conditions

Les boucles et conditions sont essentielles à chaque langage de programmation, ainsi le joueur sera en mesure de placer dans son programme un certain nombre de boucles en plus de ses actions et pourra choisir d'imposer ou non des conditions à ces dernières.

Le langage comporte les connecteurs logiques `if` et `else`. Ainsi que les boucles `while` et (`for`).

Le contenu d'un **bloc d'instructions** commence par un des connecteurs précédent et termine par une ligne vide. À l'intérieur, chaque instruction doit être indentée d'au moins un espace.

L'utilisateur peut choisir parmi une liste de **mots-clés** décrivant les différents éléments du jeu. Il peut les comparer grâce aux comparateurs `=`, `>` et `<`.

Il dispose également des booléens `true` et `false`.

Exemple de programme :

```
while voie_est_libre(haut) = true :
    avancer()

if arbre_a_cote() = true :
    couper_bois(3)
    ramasser(3)

if voie_est_libre(gauche) = true :
    tourner_a_gauche()
```

Dans le but de familiariser l'utilisateur aux langages de programmation existants, le langage utilise des connecteurs logiques en anglais. Ils seront donc introduit grâce à des vidéos explicatives en français pour en faciliter la compréhension.

6 Gestion des sauvegardes

6.1 Sauvegarde de la progression

Durant la progression de l'histoire, l'utilisateur ne va pas modifier de la même manière sa carte, en effet, si il doit trouver du bois il ne coupera pas forcément le même arbre d'une partie à l'autre. En revanche, les quêtes sont identiques d'une partie à l'autre il suffit donc d'associer à chaque session :

— L'état de la carte.
— L'état de l'inventaire.
— Le numéro de la quête en cours.

Ces informations sont enregistrées dans un fichier nommé par le nom de la session.

À chaque quête réalisée, la sauvegarde de la session est mise à jour ainsi que la progression qui est sauvegardée automatiquement.

6.2 Sauvegarde d'un programme

L'utilisateur a la possibilité de sauvegarder et d'ouvrir les programmes qu'il aura écrit.

Ils sont stockés en format texte dans un fichier nommé par l'utilisateur.

7 Compilation et exécution

Pour compiler CL4 il suffit d'exécuter :

```
make
```

Pour lancer le `launcher` :

```
make run
```

Abbreviations

CL4 CodeLab4 - L'aventure

GUI Graphical User Interface