Contents

I Historique **3**

1 Tracabilité **4**
1.1 Versions du document actuel 4

II Introduction **5**

1 Description du projet **6**
1.1 Synopsis 6

2 Documents relatifs et sources **7**

3 Objectifs **8**

4 Equipe de développement **9**
4.1 Logo 9
4.2 Membres 11

III Organisation **12**

1 Organisation de l'équipe de développement **13**

2 Rôles et responsabilités **14**

IV Exigences **15**

1 Critères pour valider une exigence **16**
1.1 Liste des critères 16

2 Exigences majeures du projet **17**
2.1 Niveaux du jeu 17
2.1.1 Niveaux projetés par la machine 17
2.1.2 Niveaux dans le monde réel 18

3 Exigences mineures du projet **19**

4 Historique des propositions d'exigences **20**

V Recherche et développement **21**

1 Game design **22**

2 Level design **23**

3 Chara design **24**
3.1 Robot (Joueur) 24

VI Réalisation 25

1 Processus **26**

2 Planning **27**

VII Suivi et contrôle 28

1 Environnement de développement **29**
1.1 Map Editor 29

2 Tests **31**

3 Optimisations **32**
3.1 Maitriser le nombre de sprites l'écran 32
3.1.1 Version non optimisée 32
3.1.2 Propositions d'optimisations 33
3.1.3 Conclusion 34

Part I

Historique

Chapter 1

Tracabilité

1.1 Versions du document actuel

Versions	Date	Commentaire
0.1	20/09/17	Création du CdC
0.2	20/09/17	Mise jour du chapitre "Optimisation"
0.3	20/09/17	Mise jour des chapitres "Description du projet" et "Exigences majeures du projet"
0.4	21/09/17	Mise jour du chapitre "Description du projet"

Part II

Introduction

Chapter 1

Description du projet

1.1 Synopsis

Un jeune robot mélancolique ne reoit plus de nouvelles de son ami denfance. Il avait pour habitude de lui envoyer des cartes postales de ses vacances mais a subitement arrté de le faire il y a peu.

Le jeune robot parcourt souvent ces cartes en se remémorant son ami qui ne donne plus de nouvelles. Il a construit une machine pouvant projeter ces cartes sur une grande toile afin de pouvoir ressentir la joie de voyager travers le monde, comme son ami a toujours pu le faire.

Au fil de ces cartes, il se rendra compte que son ami a été enlevé par un tre mystérieux et continuera de parcourir le monde grce se machine afin de continuer son enqute.

Une fois quil aura compris qui est le kidnappeur et o il cache son ami, il sy rendra, mais privé de ses pouvoirs il ne pourra rien faire. Son ami denfance fournira alors la dernière carte postale quil na pas pu envoyer. Cela permettra au jeune robot de débloquer le dernier monde et de sauver son ami avant quil ne se fasse kidnapper.

Chapter 2

Documents relatifs et sources

Chapter 3

Objectifs

L'objectif principal du projet est de proposer une expérience rétro aux joueurs.

Chapter 4

Equipe de développement

4.1 Logo

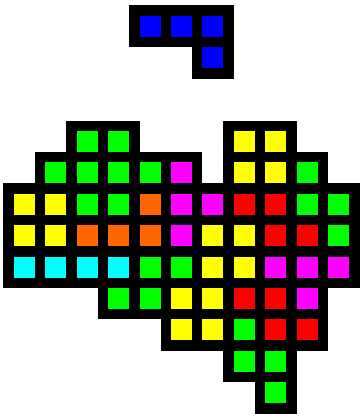


Figure 4.1: 1ère version du logo

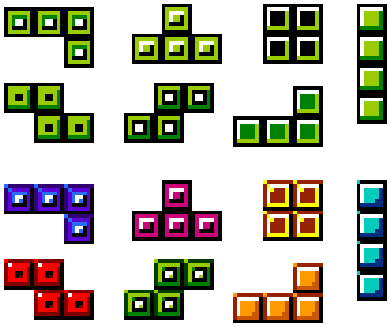


Figure 4.2: Liste des pièces utilisées

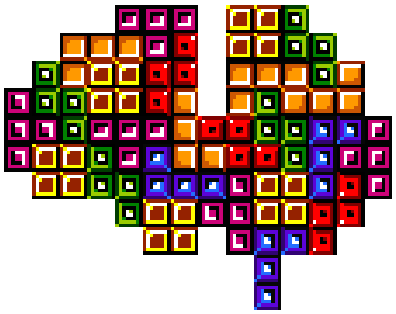


Figure 4.3: 2ème version du logo

Le logo doit représenter le potentiel de l'équipe pouvoir créer du gameplay émergent et unique. Le but est donc d'imager l'imagination et la technique =¿ Un cerveau dont la partie manquante viendrait d'ajouter. "The missing part" est pour l'instant le nom retenu pour l'équipe.

Figure 4.4: Forme finale du logo

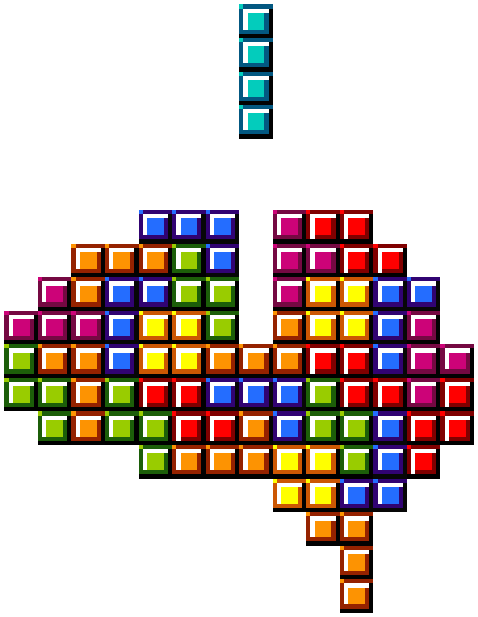


Figure 4.5: Revision des pièces

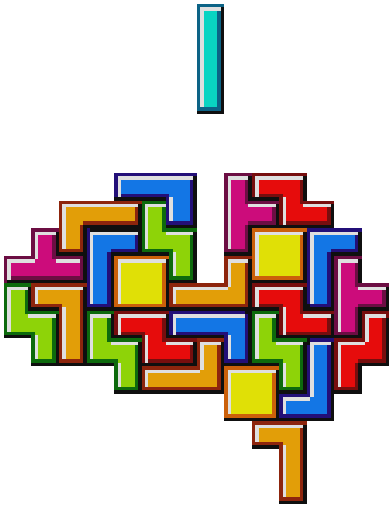


Figure 4.6: Révision des contour

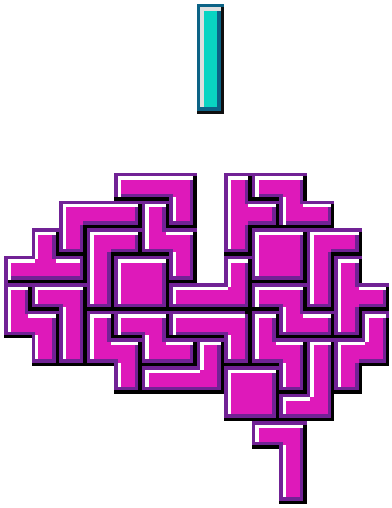


Figure 4.7: Ajustement colorimétriques

4.2 Membres

- Jean-Franois MIR
- Charlotte JUDON
- Ayoub BOUDEBIZA

Part III

Organisation

Chapter 1

Organisation de l'équipe de développement

Chapter 2

Rôles et responsabilités

Part IV

Exigences

Chapter 1

Critères pour valider une exigence

1.1 Liste des critères

Ci-dessous la liste restraintes des critères devant tre validés pour qu'une nouvelle exigence soit analysée:

- Rétro / Pixel art style.
- 8/16 bit (couleurs + images/sprites).
- Mécanique et/ou électronique (pas d'organique).
- Poétique (implique peu / pas de texte l'écran).

Chapter 2

Exigences majeures du projet

2.1 Niveaux du jeu

Les niveaux du jeu sont découpés en 2 catégories : les niveaux projetés par la machine et ceux dans le monde réel.

2.1.1 Niveaux projetés par la machine

Les niveaux projetés par la machine permettent au jeune robot de pouvoir revivre les voyages de son ami travers le monde et de faire avance le scénario (lenqute). Il y a en tout 7 mondes visiter:

- Pékin (Chine).
- Memphis (Egypte).
- Rome (Italie).
- Tenochtitlan (Mexique).
- Atlantis (sous leau).
- Ple Nord (Ple).
- Cratères (Lune).

Chacun de ces mondes est construit de la manière suivante :

- 2 niveaux de plateforme classique.
- 1 mini-boss de monde.
- 2 niveaux de plateforme classique.
- 1 boss de monde.

Les 2 premiers niveaux permettent au joueur de découvrir la mécanique principale du monde (par exemple : sauts contre les murs) et de la mettre lépreuve face elle (par exemple : les ennemis peuvent sauter contre un mur ou de sy accrocher. Le joueur ne peut pas le faire avant davoir vaincu le mini-boss. Les 2 niveaux suivants ainsi que le boss de monde mettent le joueur lépreuve avec cette mécanique (o il doit progresser avec ce nouveau pouvoir). Dans lexemple cité, le boss de monde fournirait alors le saut de mur infini.

De manière générale, le mini-boss fournit la 1ère partie du pouvoir débloquer. Le boss de monde fournit la seconde.

2.1.2 Niveaux dans le monde réel

Les niveaux dans le monde réel permettent de récupérer les cartes postales. Le monde réel prend place dans la ville natale du jeune robot et les cartes sont dispersées dans les sous-sols du hangar.

Le hangar est construit ainsi :

- 6 directions prendre au départ.
- 1 niveau puzzle dans chaque direction.
- 1 boss puzzle la fin de chaque niveau.

Le boss fourni la carte postale qui débloque laccès un monde.

Chapter 3

Exigences mineures du projet

Chapter 4

Historique des propositions d'exigences

Part V

Recherche et développement

Chapter 1

Game design

Chapter 2

Level design

Chapter 3

Chara design

3.1 Robot (Joueur)

Figure 3.1: évolution du chara design du robot

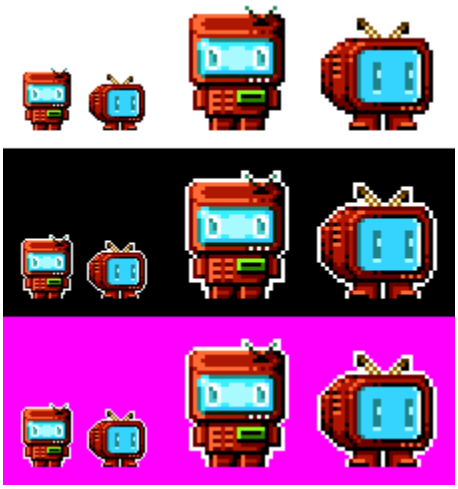
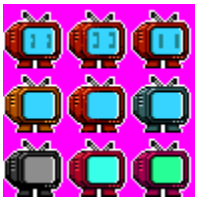


Figure 3.2: Tests de colorimétrie



Part VI

Réalisation

Chapter 1

Processus

Chapter 2

Planning

Part VII

Suivi et contrôle

Chapter 1

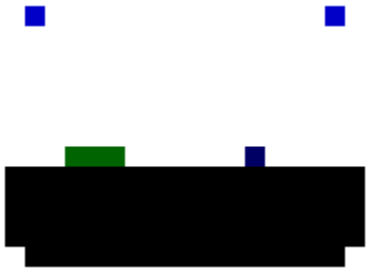
Environnement de développement

1.1 Map Editor

L'éditeur de niveau (minimap) se présente de la manire suivante:

- Une image éditable dont chaque pixel représente un bloc de 32x32 pixel l'écran.
- Un interpréteur codé en Python qui transcrit les couleurs des pixels en ID.
- Un processus codé en Python également qui contruit le sprite partir de l'ID.

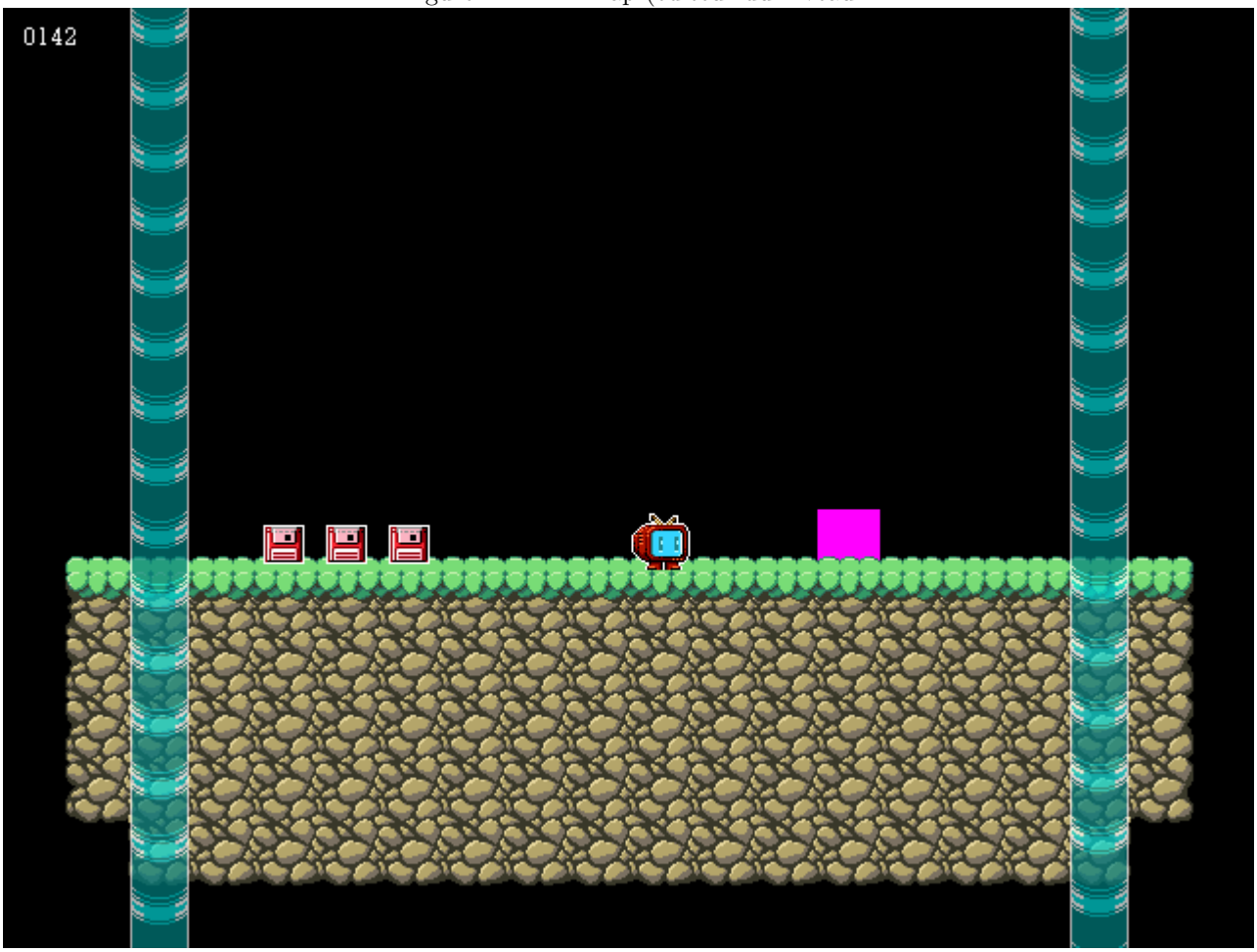
Figure 1.1: Minimap (éditeur du niveau



Les couleurs des pixels de la minimap codent donc les ID des blocs (sprites) de 32x32 pixels. La rgle est la suivante:

- Chaque valeur R,G et B de la couleur du pixel représente un bit de codage.
- Les bits sont codé en base 3 (valeurs possible par bit : 0, 1 ou 2).
- les valeurs R, G et B sont multipliées par 100 (0, 100 ou 200) afin de les différencier lors de l'édition
- Exemple [R;G;B]: [0;200;100] devient [0;2;1]. L'ID = 0*9 + 2*3 +1*1 = 7.

Figure 1.2: Minimap (éditeur du niveau



Chapter 2

Tests

Chapter 3

Optimisations

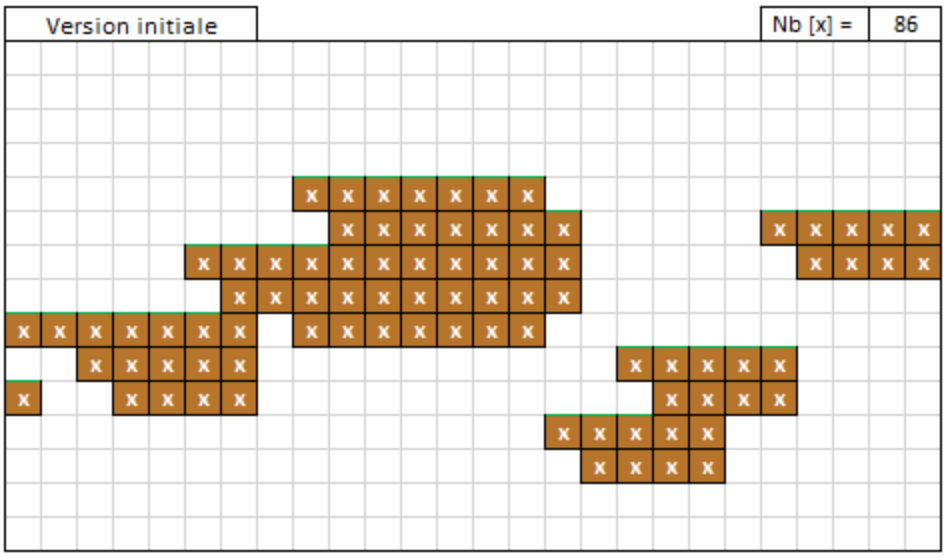
3.1 Maitriser le nombre de sprites l'écran

Les sprites sont le coeur mme d'un jeu vidéo. Il est primordial de maitriser leur rafraichissement et leur nombre pour ne pas se retrouver avec de grosse perte de FPS.

3.1.1 Version non optimisée

Initialement, le projet est programmé ainsi : Une grille sur laquelle viennent se poser des sprites (aka tiles) de collision :

Figure 3.1: Mapping initial des tiles sur une grille

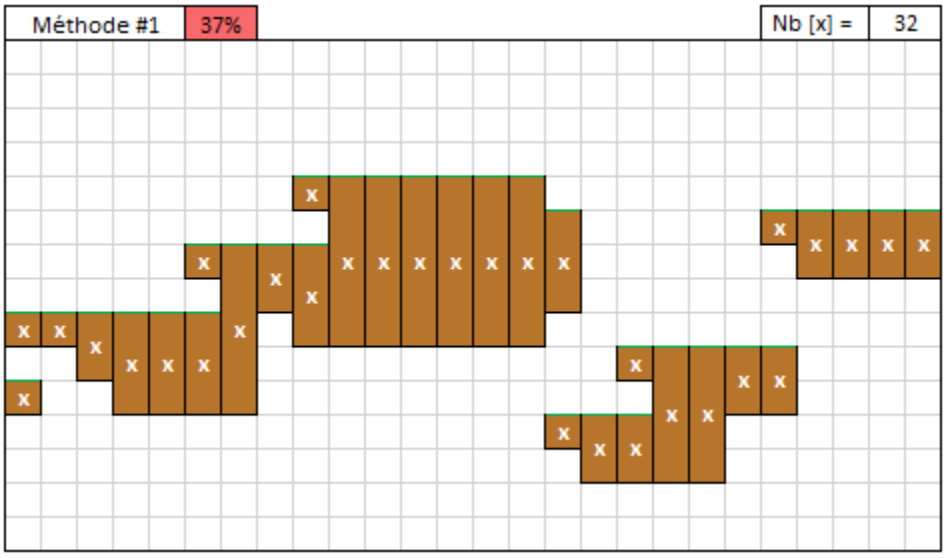


Sur cette 1ère vue on constate un grand nombre de tiles pour former les plateformes du jeu. Chaque tile nécessite un rafraichissement chaque frame pour déterminer si il se trouve l'écran ou non (afin de l'afficher et de le considérer dans la liste des tiles de collision actuel).

3.1.2 Propositions d'optimisations

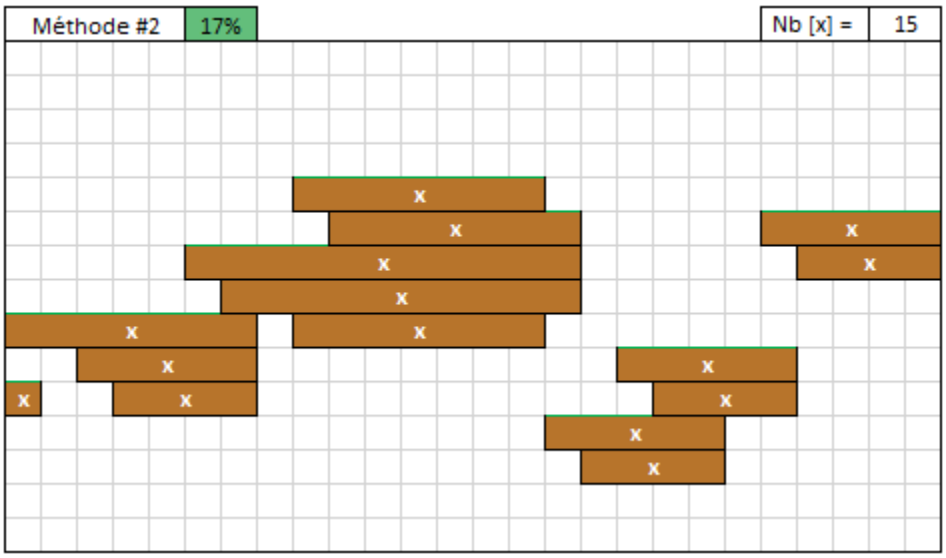
La réduction de ces tiles en les regroupant permet d'accélérer leur rafraichissement et de réduire d'autant les check de collisions. Ci-dessous sont présentées 3 méthode distinctes qui détaillent chacune le nombre de tiles restant ainsi que la part qu'elle représente face au mapping initial (le code couleur compare leur efficacité):

Figure 3.2: Mapping optimisé N1



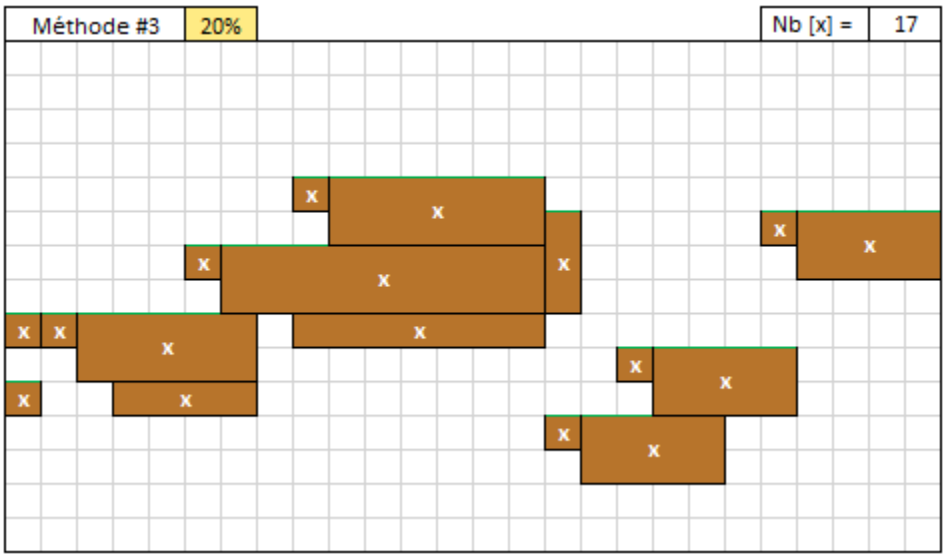
Chaque tile regroupe ici ceux verticalement adjactents. Le nombre de tiles est divisé par 3 sur ce 1er essai. Il limite l'affichage de ceux présent dans l'écran uniquement mais leur nombre augmentera linéairement en fonction de la longueur du niveau générer.

Figure 3.3: Mapping optimisé N2



Chaque tile regroupe ici ceux horizontalement adjactents. Le nombre de tiles est divisé par 6 sur ce 2ème essai. Il y a bien moins de tiles manipuler et rafraichir mais cela peut générer de très long tiles (plus long que l'écran) maintenir afficher entièrement.

Figure 3.4: Mapping optimisé N3



Chaque tile regroupe ici ceux verticalement et horizontalement adjactents. Cette opération donne de bon résultats (similaire la méthode N2) mais son implémentation semble difficile réaliser et maitriser.

3.1.3 Conclusion

Suite au différents tests réalisées, on peut synthétiser les propositions ainsi:

Versions	Nb tiles	% tile utilisés	Intérts	Contraintes
Version initiale	86	100%	Implément.n simple	Très lourd rafraichir
Group. vertical	32	37%	Aucun tile hors écran	Pas efficace sur longs niveaux
Group. horizontal	**15**	**17%**	**Très peu de tiles**	**Tiles plus long que l'écran**
Group. vert. + hor.	17	20%	Très peu de tiles	Difficile implémenter

La proposition d'optimisation retenue est la seconde : Regroupement des tiles horizontalement.

List of Figures

4.1 1ère version du logo 9
4.2 Liste des pièces utilisées 9
4.3 2ème version du logo 9
4.4 Forme finale du logo 10
4.5 Revision des pièces 10
4.6 Révision des contour 10
4.7 Ajustement colorimétriques 10

3.1 évolution du chara design du robot 24
3.2 Tests de colorimétrie 24

1.1 Minimap (éditeur du niveau 29
1.2 Minimap (éditeur du niveau 30

3.1 Mapping initial des tiles sur une grille 32
3.2 Mapping optimisé N1 33
3.3 Mapping optimisé N2 33
3.4 Mapping optimisé N3 34