

Projet Ninja Saga

Moïse BEUGRE – Ouajih DADAOUA



Figure 1 - Exemple du jeu Ninja Saga

Table des matières

1 Objectif	3
1.1 Présentation générale	3
1.2 Règles du jeu	3
1.3 Conception Logiciel	3
2 Description et conception des états	4
2.1 Description des états	4
2.2 Conception logiciel	4
2.3 Conception logiciel : extension pour le rendu	4
2.4 Conception logiciel : extension pour le moteur de jeu	4
2.5 Ressources	4
3 Rendu : Stratégie et Conception	6
3.1 Stratégie de rendu d'un état	6
3.2 Conception logiciel	6
3.3 Conception logiciel : extension pour les animations	6
3.4 Ressources	6
3.5 Exemple de rendu	6
4 Règles de changement d'états et moteur de jeu	8
4.1 Horloge globale	8
4.2 Changements extérieurs	8
4.3 Changements autonomes	8
4.4 Conception logiciel	8
4.5 Conception logiciel : extension pour l'IA	8
4.6 Conception logiciel : extension pour la parallélisation	8
5 Intelligence Artificielle	10
5.1 Stratégies	10
5.1.1 Intelligence minimale	10
5.1.2 Intelligence basée sur des heuristiques	10
5.1.3 Intelligence basée sur les arbres de recherche	10
5.2 Conception logiciel	10
5.3 Conception logiciel : extension pour l'IA composée	10
5.4 Conception logiciel : extension pour IA avancée	10
5.5 Conception logiciel : extension pour la parallélisation	10
6 Modularisation	11
6.1 Organisation des modules	11
6.1.1 Répartition sur différents threads	11
6.1.2 Répartition sur différentes machines	11
6.2 Conception logiciel	11
6.3 Conception logiciel : extension réseau	11
6.4 Conception logiciel : client Android	11

Objectif

1.1 Présentation générale

L'objectif de ce projet est la réalisation du jeu "Ninja Saga", avec les règles les plus simples. Un exemple est présenté en Figure 1. Des combats entre personnages pourront être faits en tenant compte des points de vie, des points de mouvement et des points d'attaques en possession selon le personnage choisi.

Les personnages auront des caractéristiques qui influencent le combat, par exemple plus un personnage sera fort grand et imposant moins il sera rapide mais ses attaques auront plus d'impact.

1.2 Règles du jeu

Le joueur choisit un personnage en début de partie pour faire un combat. Le joueur gagne lorsque son adversaire n'a plus de points de vie et perd lorsqu'il perd tous ses points de vie.

1.3 Conception Logiciel

L'affichage repose sur les textures suivantes:

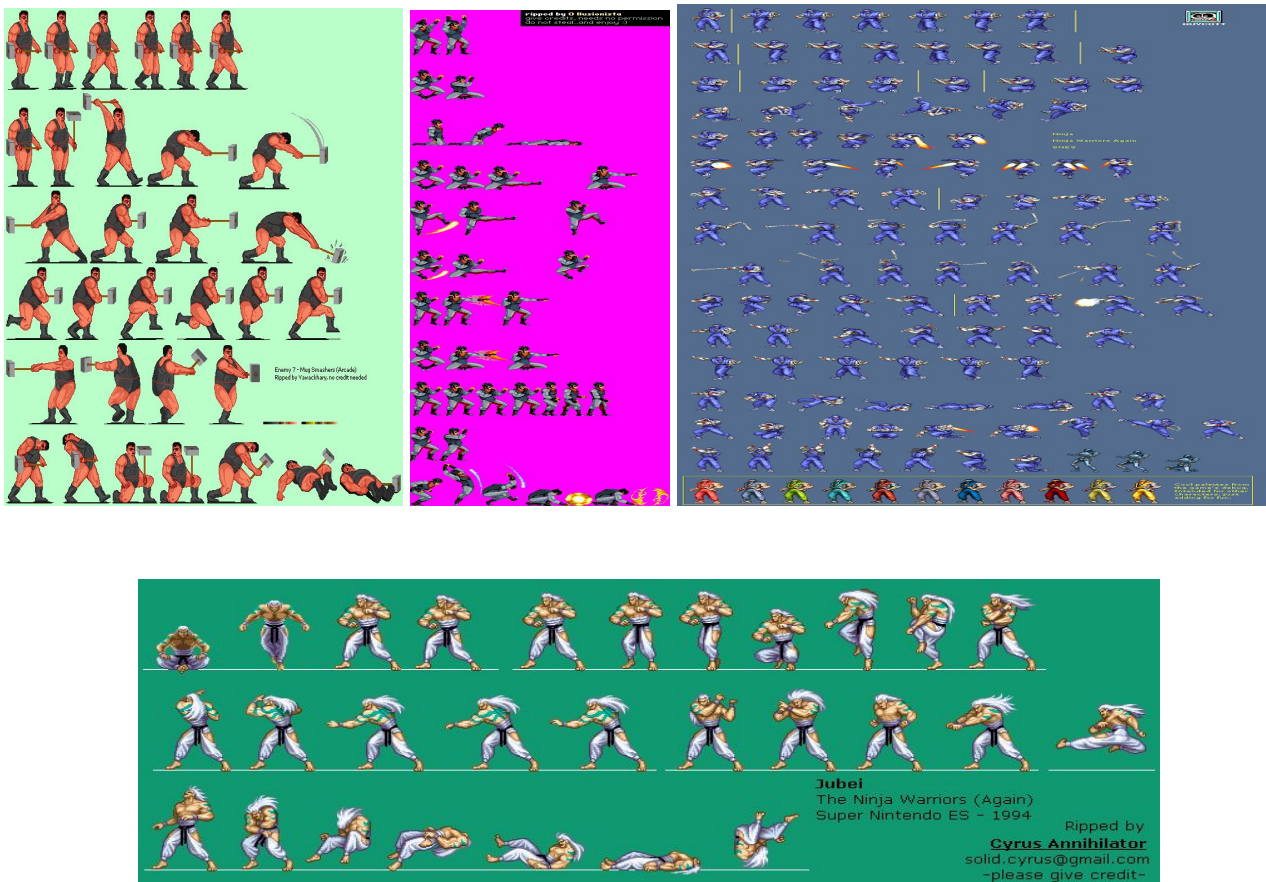


Figure 2 - Textures pour les personnages

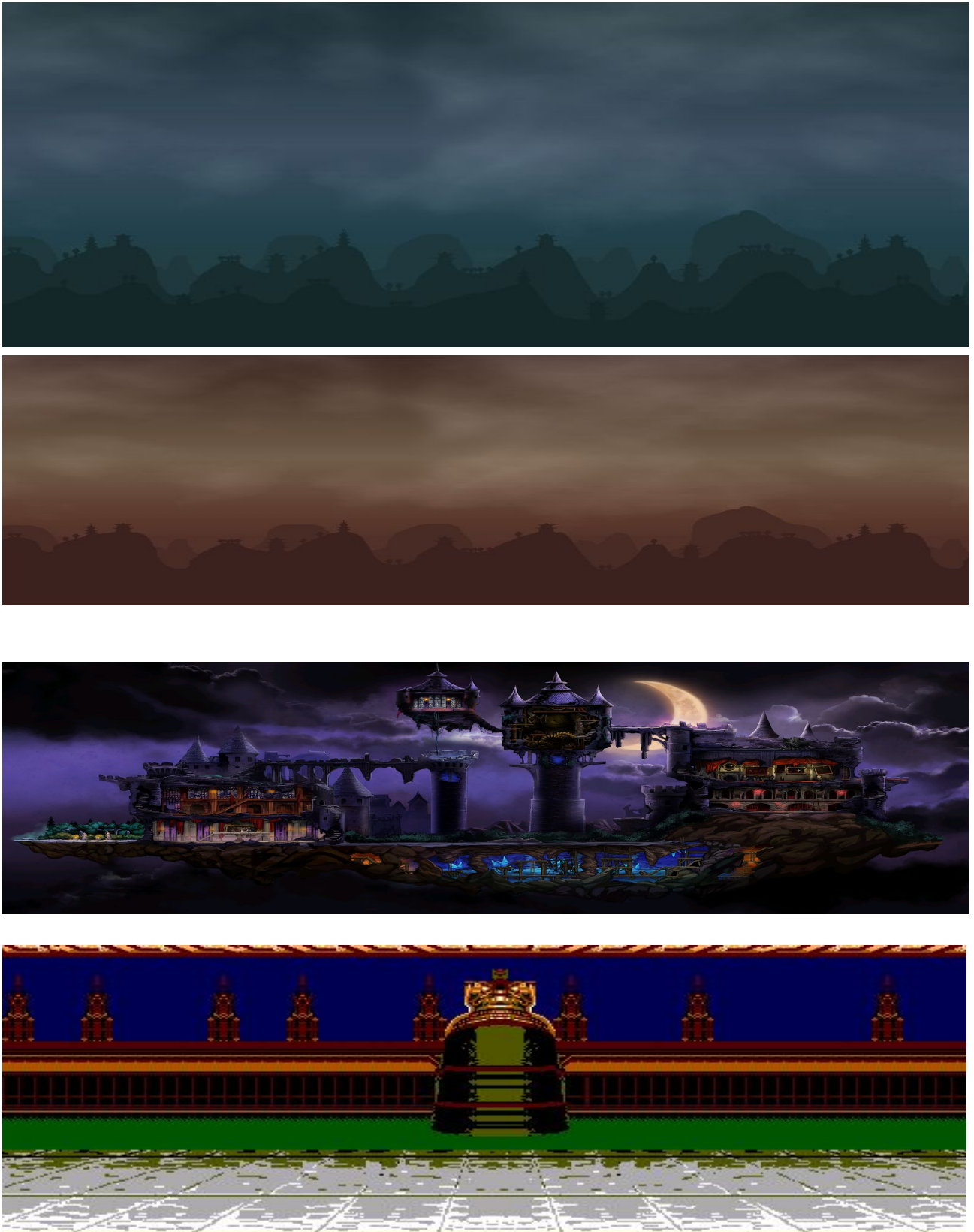


Figure 3 - Textures pour les terrains de combat



Figure 4 - Textures pour les tuiles de lettres et chiffres

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très apprécié. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

Illustration 1: Diagramme des classes d'état

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémentent pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects liés au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est une parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation

