« Le monde des ronrons »

TECHNICAL DESIGN

Christophe Kohler Fev 2006

Table des matières

Outils

“Le monde des ronrons” est un jeu qui contient peu de données. Il y a 5 créatures et un décor de 15 écrans de long. Il y a 5 mini jeux .

Génération des données

Toutes les données 3D seront générées avec un éditeur 3D. MAYA 7 est choisi.

Les objets et données seront exportés depuis MAYA en utilisant le format COLLADA (libre et gratuit). C’est un format simple, ascii (xml exactement) qui est facile à relire.

Dans MAYA seront crées les décors, les créatures, les animations et les limites d’aires.

Code

Les compilateurs utilisés dépendront de la plateforme :

- PC : Visual Studio 2005 Express (gratuit)

- PS2 : gcc (gratuit) ou CodeWarrior 3.6 (acheter licence).

Données graphiques

Résultat final souhaité a l’écran

Le but de ce chapitre est de définir a quoi doit ressembler le rendu final du jeu.

Il a déjà été décidé de resté simple, dans un style très « dessins animé ». Des aplats de couleurs seront utilisés, et des détails seront ajoutés pour rendre le tout plus riche.

Voici un exemple d’herbe :

Le fond est rempli d’une seule couleur et des touffes donne l’information que cela représente de l’herbe.

Dans les jeux on utilise d’habitude des textures détaillées comme celle ci :

Mais cela ne correspond pas à ce que l’on veut. Il faut que les pixels soient les moins visible possibles.

Les personnages auront de fines lignes noires sur les contours pour souligner les volumes. Comme sur le dessin suivant :

Liste des éléments de décor

Le décor est composé de plusieurs éléments :

Surface de sol : Herbe, terre, sable, eau

Détails du sol : Fleurs (sur herbe, eau), petits cailloux (terre, sable), des souches d'arbres (terre et herbe), des trous (terre)

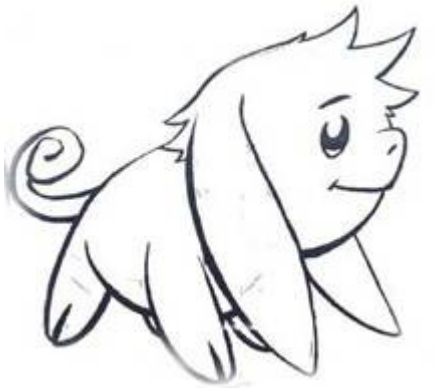
Transition entre les type de sols : Herbe <-> Terre, Herbe <-> Eau, Sable <-> Eau

Objets : Arbres, toilettes, douche, barrière, gros rochers, grosses fleurs, nourriture, pont, toboggan, balançoire.

Liste des personnages

Voici les 5 sortes de personnages

Créature 1



Il bouge sur ses 4 pattes.
Son corps est lisse.
Il a un nez qui dépasse de son visage
Ses oreilles peuvent balancer un peu

Créature 2

Il bouge sur ses 4 pattes.
Son corps est poilu
Il a un petit nez et des joues rondes.

Créature 3

Quand il est au repos il est sur ses 4 pattes.
Quand il avance, il flotte dans les airs.
Son corps est lisse.
Il a un gros nez.
Il peut bouger ses oreilles.

Créature 4

Il bouge sur ses 4 pattes (un peu en rampant)
Il est très poilu
Il a un gros nez et des joues rondes.

Créature 5

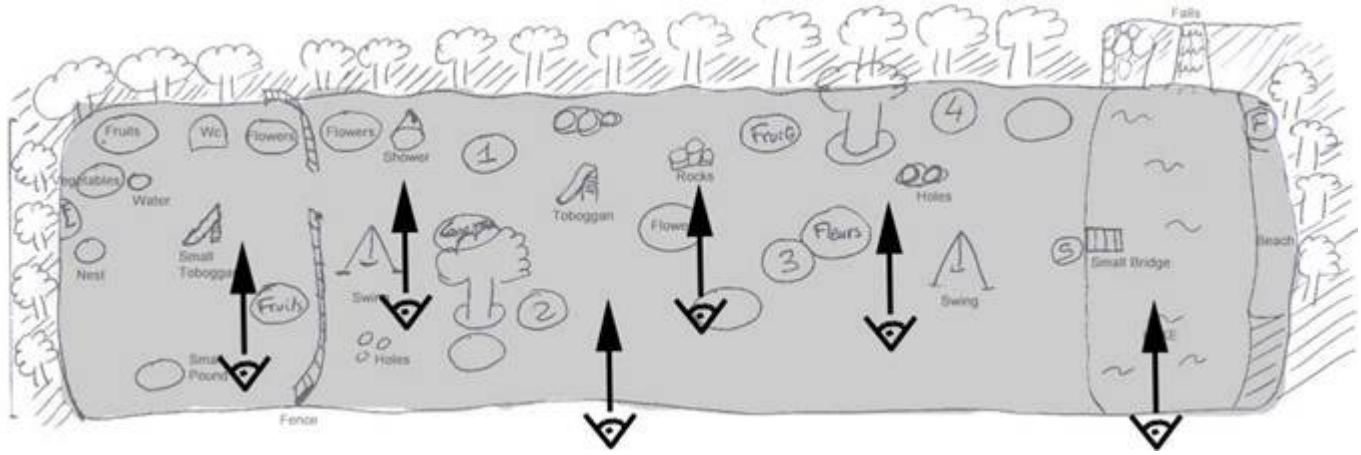
Il marche sur ses 2 pieds comme un humain.
Son corps est lisse. Il des cheveux sur l’arrière de sa tête.
Il a un très gros nez.
Il a une grande bouche.

Techniques de rendu

Le but de ce chapitre est de définir comment les éléments définis dans le chapitre précédent vont être rendu à l'écran.

Vue

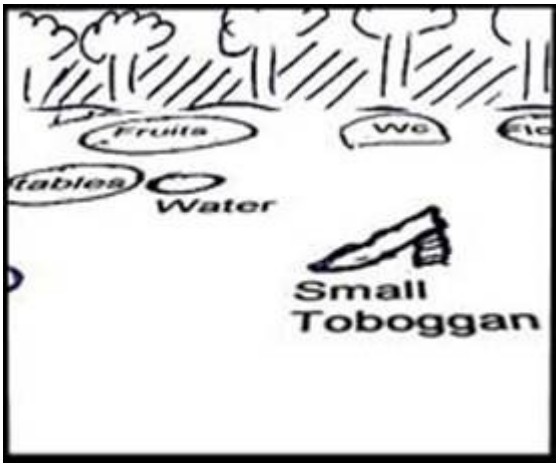
Un point important du jeu est que la camera ne tournera pas. Elle se déplacera de droite a gauche et d'avant en arrière. Le schéma suivant montre quelques position possible de la camera :



Ceci va aider à faire certains choix techniques.

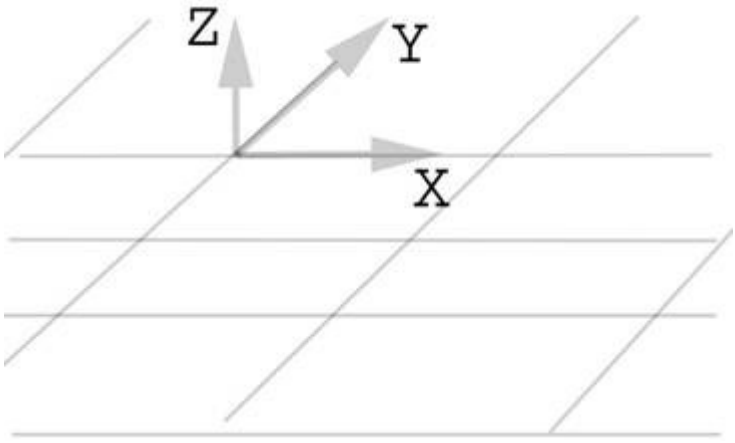
La contrainte définis dans le "game design" est que la vue doit montrer l'équivalent de 2 « unités écran » de décor. Le décor est profond de 4 écrans, donc la camera doit toujours montrer la moitié de la profondeur du décor.

La camera est toujours pointé vers le nord. Elle est a 5 mètres au dessus du sol (1 mètre = une unité dans le décor), et elle pointe légèrement vers le sol. Ce qui donne une représentation qui est vaguement comme ceci :



Décor

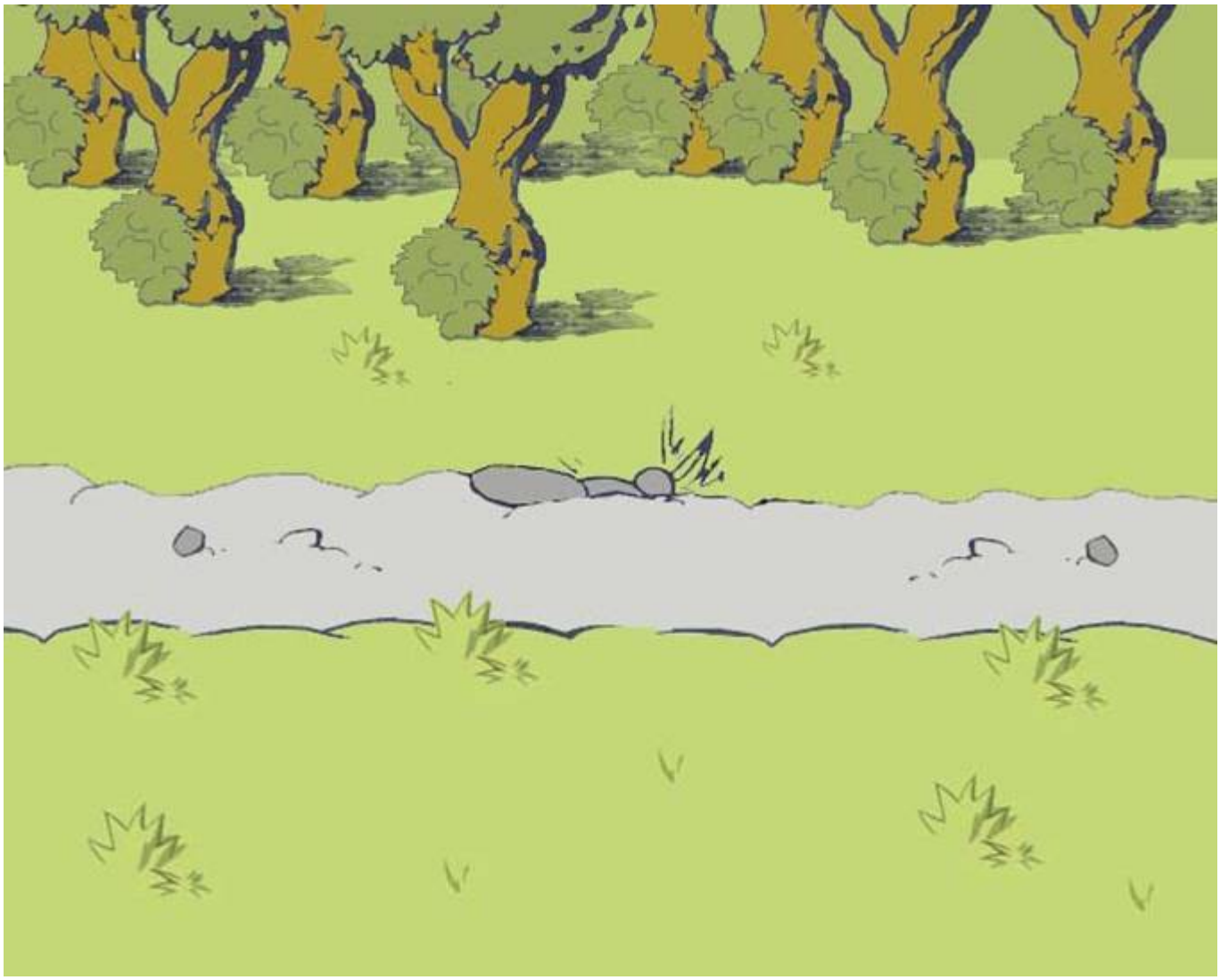
Le décor peut être considéré comme une carte en deux dimensions. Nous utiliserons les coordonnées X et Y pour nous repérer sur le sol. Z sera l'altitude. Le repère est donc le suivant :



C'est un repère "main droite"

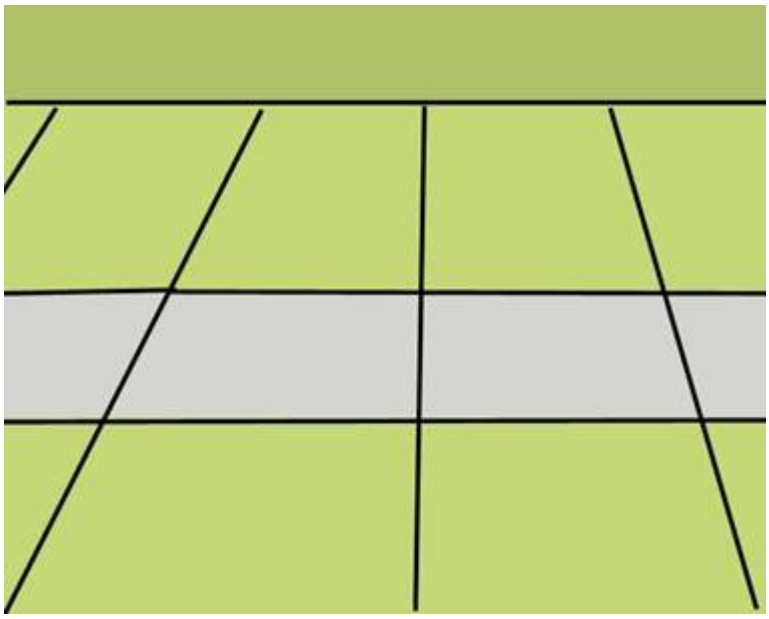
Le fait que la camera ne tourne jamais nous permet d'utiliser une technique intéressante. Nous allons utiliser un décor en 3D mixé avec des éléments en 2D (appelés « billboards »). Ceci nous permet d'avoir plus de détails avec moins de polygones. Cela permets aussi de faire disparaître l'effet « on voit des pixels » des objets finement modélisés et texturés.

Exemple de représentation :



Sur cette image, le sol est en 3D. Par-dessus on ajoute des éléments en 2D (des faces qui sont toujours face a la camera).

Sur l’image précédente, le sol est comme ceci :



Ce sol est obtenu en affichant un objet 2D texturé avec des images en aplat (en gros des textures avec une seule couleur). On pourra ajouter du détail dans ces textures si il faut.
(les lignes noires ici sont ajoutés pour montrer l’aspect 3D, mais ne sont pas affichées dans le jeu)

Sur cette base, on ajoute des billboards qui sont des images simple (des « barrière »). Ces barrière ont une position dans le monde 3D et sont orientés vers la camera.

Voici les éléments qui ont été utilisés pour constitué l’image précédent : (les parties blanches sont transparentes)

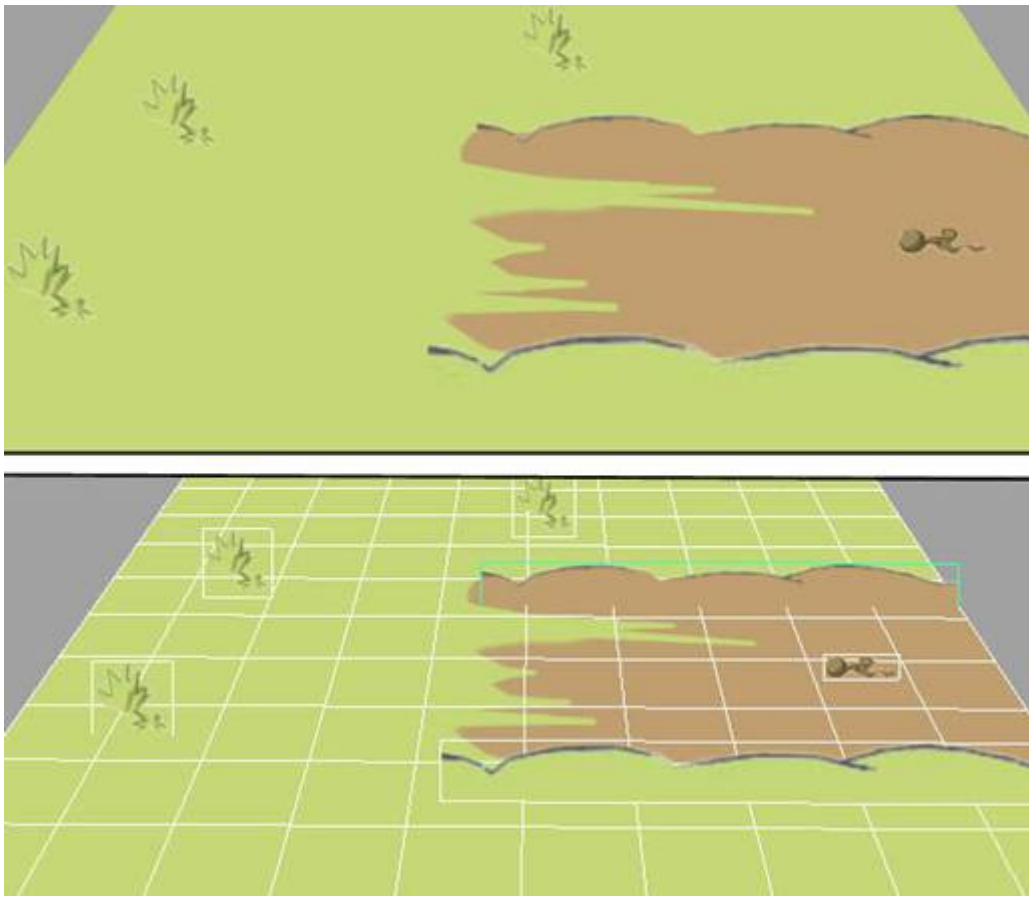
Avec ces images, on peut obtenir des choses très détaillées. Ces images sont bien plus jolies que de vrais objets modélisés et texturés. Cela va nous permette d’obtenir un monde « mignon ».

Eléments verticaux.

La technique des billboards fonctionne bien avec des éléments horizontaux (comme le chemins). Pour des éléments verticaux (c'est-à-dire qui s’étende du devant vers le fond), il faut utiliser une technique différente.

Pour le sol, les transitions dans le sol seront incluses dans les textures.

Prenons un exemple, transition herbe / terre



Les bords horizontaux du morceau de terre sont fait en utilisant des Génération. La transition verticale entre l'herbe et la terre est faite avec une texture dans le sol :

Certaines bordures (comme herbe / eau) pourront être faites en utilisant beaucoup de billboards qui se superposent.

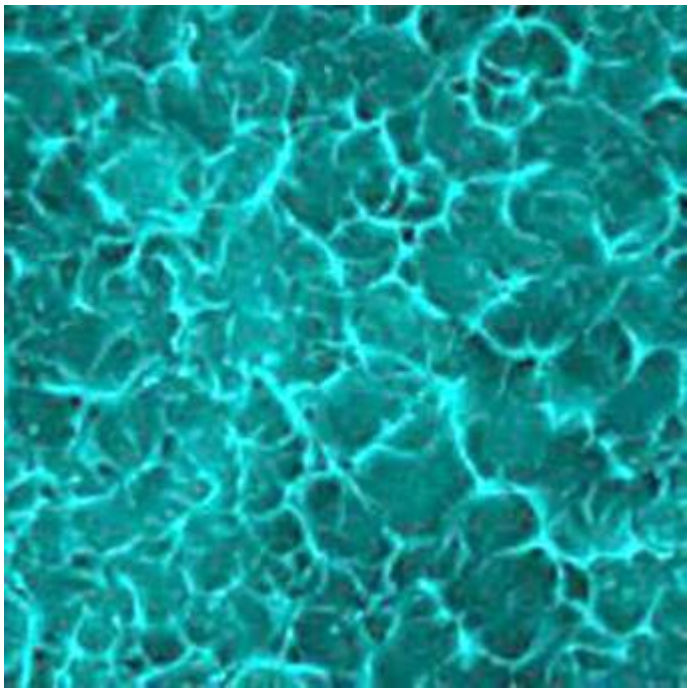
Type éléments.

Le gros travail va consister à définir quels éléments seront en vraies 3D et quels éléments seront en Génération. Ca dépend beaucoup de leur position dans le monde et de leur complexité. Les gros objets seront en 3D (comme le toboggan), certains seront un mélange des deux (comme les gros arbres, tronc 3D et feuillage en ensemble de billboards) et certains éléments seront en Génération purs comme les petits détails sur le sol.

Texture animée

Pour l’eau et la cascade, le mouvement sera crée en animant la texte. La géométrie 3D ne bougera pas, c’est la texture qui changera. On déplace les UV pour créer un mouvement.

L’eau est un plan 3D avec une texture transparente (on voit le fond de décor en dessous) comme ceci : (mais plus cartoon bien sur)



La cascade pourra utiliser une texture qui ressemble a ça : (toujours en plus cartoon bien sur)

Les textures devront “tiler” (pouvoir être répéter et bien se joindre) pour pouvoir représenter une grande surface.

Particules

Les particules sont de petites barrières qui sont utilisés pour ajouter de la vie au monde.
Par exemple des particules seront utilisées pour créer une brume d’eau en mouvement au bas de la cascade, pour les bulles des spaghettis. Quand une créature prendra sa douche, quelques bulles et de la vapeur seront émises. Ceci sera fait avec un petit bout de code simple (déplacement de barrière)

Halos

Les halos sont des barrières qui entourent les lumières. Il y en aura autours des lumières de la balançoire, cela donne un petit coté féerique. Les halos permettent de donner l’impression qu’une lumière est allumée.

Contraintes

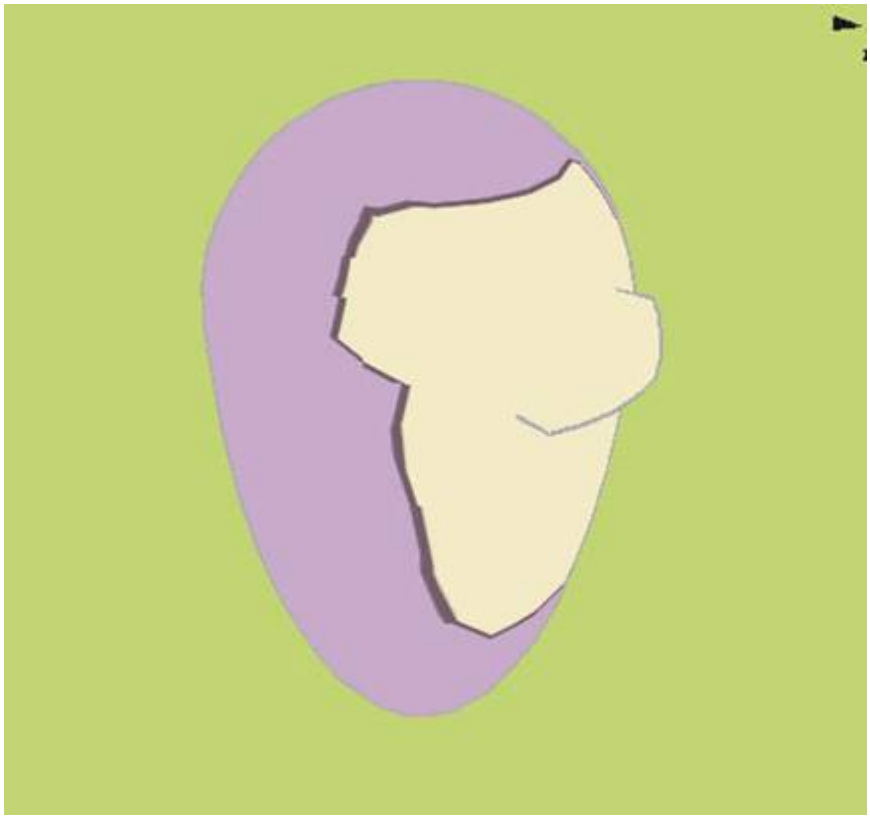
Le nombre total de polygones pour le décor est fixé a 78 000
Cela permets d'avoir un décor très léger a écran (5000 polygones) et une taille total de donnée de 4Mo a peu prêt, ce qui permet d'avoir tout le décor en mémoire (pour la PS2).

Personnages

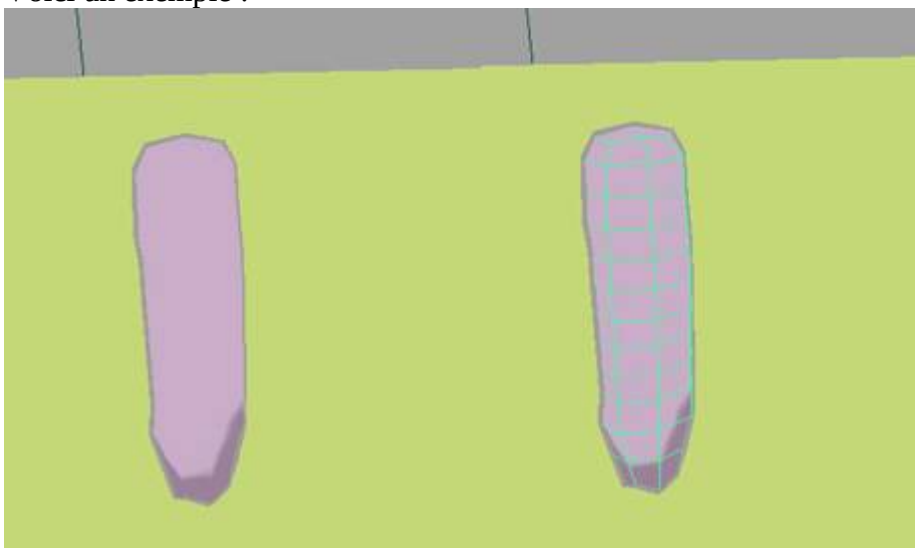
Techniques générales

Les personnages sont modélisés avec les techniques des squelettes et de la peau (Personnages « Skinés ».

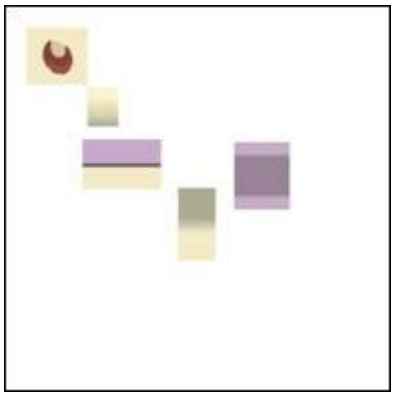
-La petit ligne noire autours des personnage ➔ Ceci peut être fait en programmation. Il faut fait un rendu en fil de fer en se mettant en mode « face arrière » (backface culling). Le dessin de la ligne peut être améliorer (non montré ici) pour obtenir un effet dessiné plus sympa.
Voici une illustration :



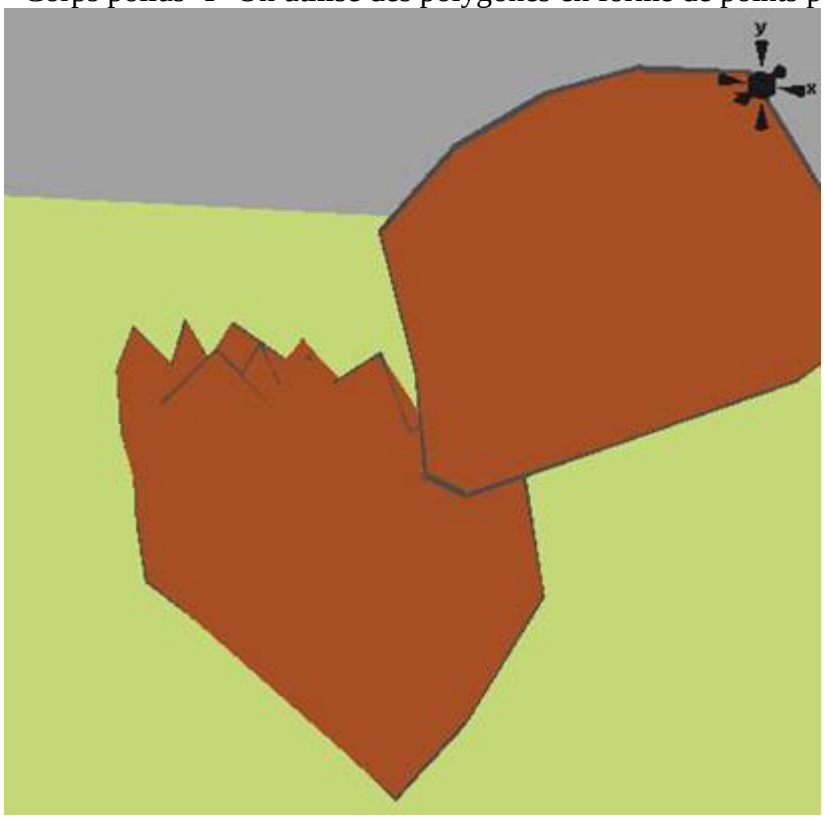
- Les ombres sur les personnages seront contenus dans la texture (pas d'éclairage temps réel).
Voici un exemple :



Ceci est fait en utilisant la texture suivant :



- Corps poilus ➔ On utilise des polygones en forme de points pour donner un effet de poil.



- Pour le personnage 3, il faut qu'il ait des poils sur ventre. Ceux ci seront dessiné dans la texture (on ne peut pas faire des polygones car les poils sont trop petits et les ligne noires de contours vont être pas belles sur cette partie).

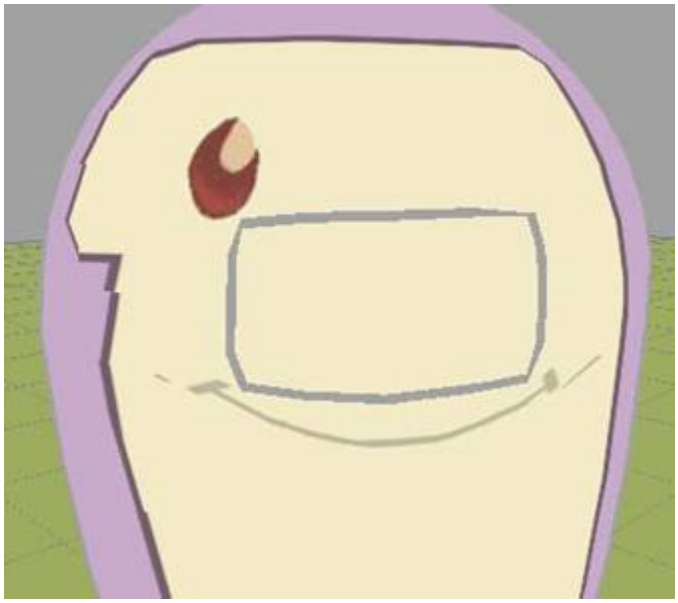
Partage de modélisation

L'artiste déterminera si il est possible de mettre en commun des squelettes (pour réduire le coût de création des animations)

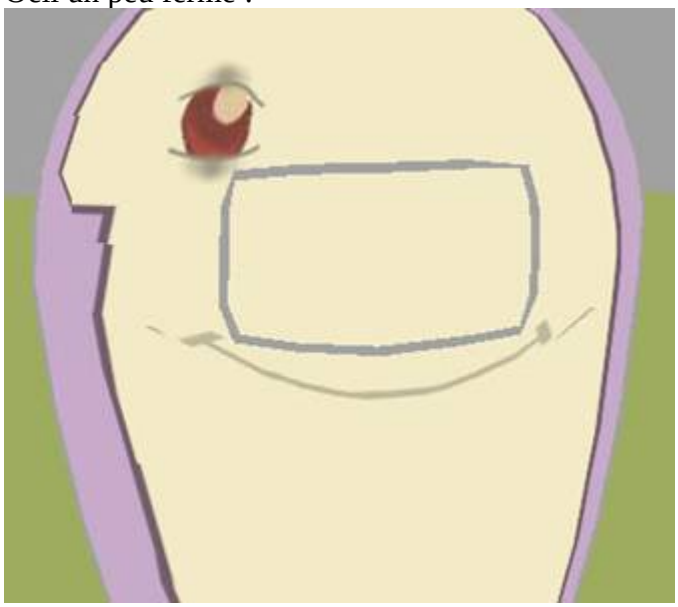
Animations du visage

Les expressions du visage seront faites en utilisant des textures (pas de modélisation).
La technique est la même que pour les dessins animés ou les images se succède pour faire un mouvement d'œil ou de bouche. On définira une dizaine d'image pour les yeux et la bouche.

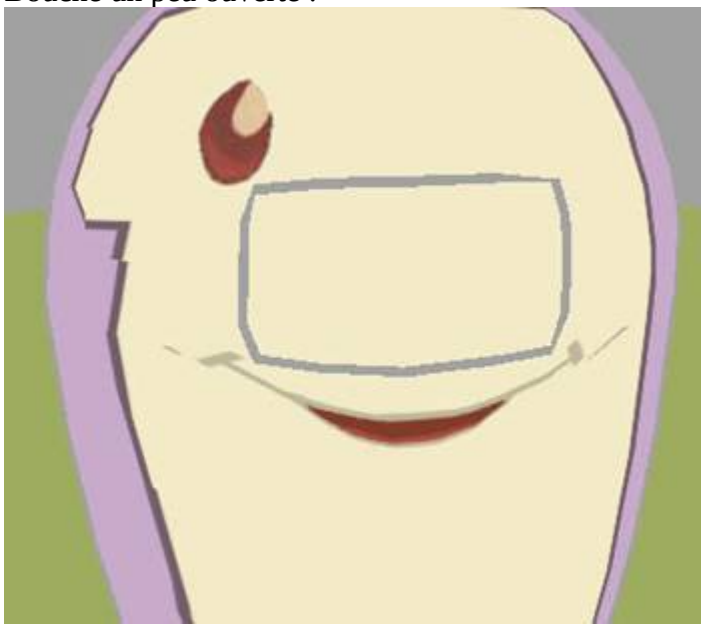
Voici un exemple :



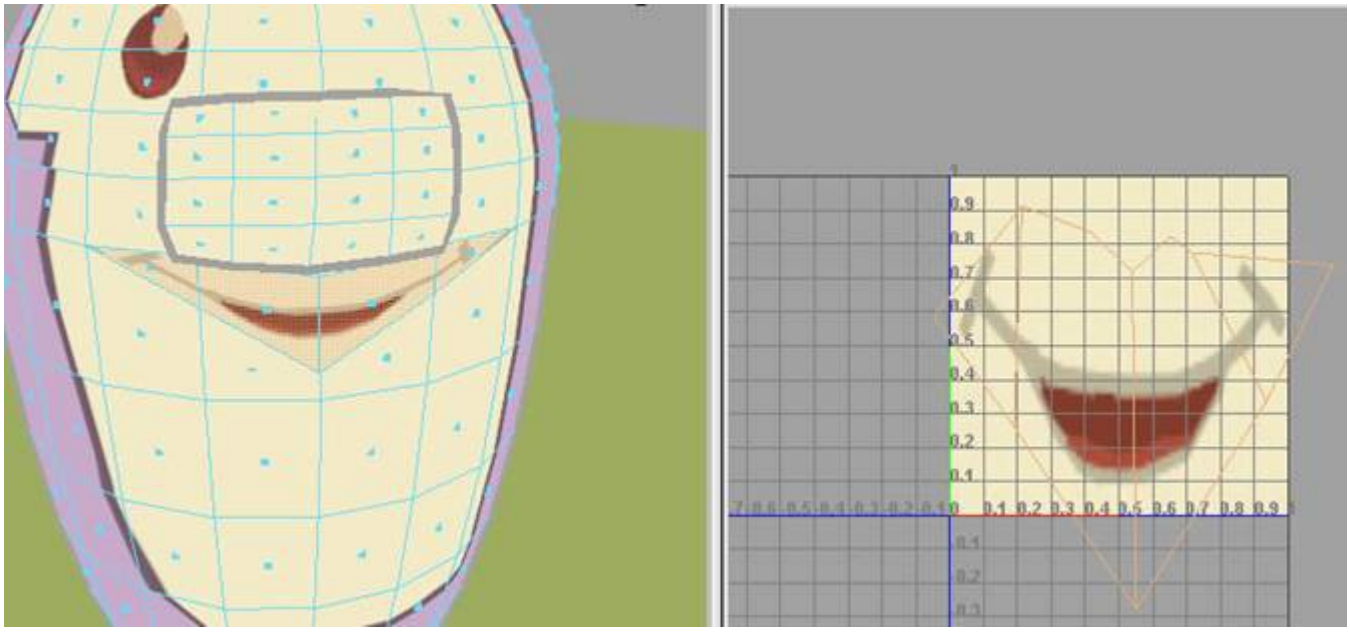
Oeil un peu fermé :



Bouche un peu ouverte :



Ceci est obtenu en séparant les texture d’oeil et de bouche au niveau du modeler et en changeant ses textures en temps réel :



Les animations faciales seront définies dans de simples fichiers textes. Un petit outil sera mis au point pour pouvoir régler facilement les animations.

Créature 1 – Techniques de modélisation et d’animations

Les oreilles sont longues et peuvent balancer. L’artiste devra animer les oreilles.

Créature 2 – Techniques de modélisation et d’animations

L’artiste devra modéliser des poils.

Créature 3 – Techniques de modélisation et d’animations

Il a un peu de poil sur le ventre. L’artiste devra inclure ça dans la texture.
Les oreilles sont rigides et peuvent battre, ce sera animé avec le squelette.
L’ombre des oreilles ne bougera pas.

Créature 4 – Techniques de modélisation et d’animations

L’artiste devra modéliser les poils
Les poils sur le visage et la queue seront dans la texture.

Créature 5 – Techniques de modélisation et d’animations

Il a une grosse bouche. Les techniques de modélisation seront les même que pour les autre personnages, donc la bouche ne pourra pas être ouverte physiquement
Il a des cheveux sur sa tête, l’artiste devra les animer avec le squelette.

Contraintes

Les contraintes globales pour les personnages sont :

800 polygones approx pour le perso.

64 os maximum, 2 influence d’os par vertex.

Détails sur le processus de création de données

Graphisme de décor

Le décor est crée sur le modeleur 3D. Tous les polygones sont texturés. Les billboards sont placés par l'artistes (une face carré = quad = un Génération). L'artiste les orientera pour obtenir le meilleur résultat dans le jeu.

Les texture sont de simple images, fait avec n'importe quel éditeur d'image (comme photoshop).

Objets animés

Seul la balançoire est animée. L'animation sera fait en code.

Personnages

Les personnages sont modélisés sous le modeleur 3D, les animations seront aussi faite avec ce modeleur (ou n'importe quel logiciel, mais au final cela doit être réimporté dans le modeleur). Les personnages sont constitués d'un squelette (avec des os) et d'une peau

Animations des personnages

Les animations seront définies avec 24 clés par images (cela peut être descendu a 15 minimum).

La liste complète des animation a produire sera définies dans une liste a part.

Interfaces

Les interfaces seront faite en code.

Menu

Les menus seront faits en code lament.

Processus pour créer les autres données

Poly lignes pour définir les zones

Les différentes zones dans le jeu sont définies en vue du dessus (2D). Ce sont des poly lignes fermées, ces zones sont :
- L'extérieur du monde (la créature ne pourra pas aller plus loin)
- Les zone de chaque objets (quand une créatures est dans cette zone, elle peut interagir avec l'objet).
- Les zones spécifiques comme les zones d'eau, les marres, le sable.Ces zone seront utilisés pour modifier le comportement de la créatures, comme par exemple lorsqu'elle est dans l'eau elle nage au lieu de marcher.
- Les zones de collisions autours des objets (arbres, barrière, ….). Les créatures ne pourront pas aller dans ces zones.

Les zones sont de simples listes de points. Le nom de la zone définira ses propriétés.

Les trajectoire des personnages non jouables (NPC)

Les personnages non jouables se déplacent en respectant des scripts prédéfinis. Les scripts sont constitués de liste de positions. Pour chaque position, on ajoute un temps d'attente et une action.

Monde physique

Le jeu doit pouvoir gérer la position des créatures et de la camera. On utilisera des données simplifiées qui définiront le monde physique. Le décor n'est pas très vaste et aucune zone n'a d'étage. La collision avec le sol peut être gérée en 2D (vu du dessus). Les altitudes seront stockées dans une grille a pas constant. Le pas de la grille sera défini pour avoir une précision suffisante.
Les zones de collision sont définies par les poly lignes vues précédent.

Contraintes et technique pour chaque plateforme

Technique pour PC

Le moteur de jeu est le “Game Incubator ». Il est déjà assez complet et il est open source donc customisable.
Tous les formats de textures sont supportés dans le GI. Le png est préféré car il ne dégrade pas. Pour optimiser la place utilisée dans la carte graphique, le format DDS sera utilisé pour certaines textures.

Textures :
- La taille doit être multiple de 2 (32,64,128,256,512,1024).
- Les textures peuvent être carrées ou rectangulaires.
- La taille des textures dépend de la taille qu’elle représente à l’écran. Si la résolution de référence est le 1280x1024 et qu’un arbre peut être vu avec une taille de moitié écran, alors la taille de la texture sera 512x512. Si un rocher peut être vu sur 1/8eme écran alors la taille de la texture sera 128x128
- Regrouper si possible les petites textures sur une grosse pour les objets similaires ou les textures composant un même objet.

Technique pour PS2

Pour l’affichage, la librairie gratuite de Sony “Librairie Graphique Haut niveau” sera utilisée.
Pour le son, le moteur “multistream” de Sony sera utilisé (il est aussi gratuit).
Pour le système de fichier, j’utilise mon system “MAL’.

Les textures seront en 256 couleurs palettisés (ou 16 couleurs autant que possible). Chaque entrée de palette peut supporter une valeur d’alpha (transparence). Le format de texture est le TIM2, il existe de petits outils gratuits pour les générer.

La taille des textures sera la moitié de celle pour PC. Les contrainte sont les même que pour PC.

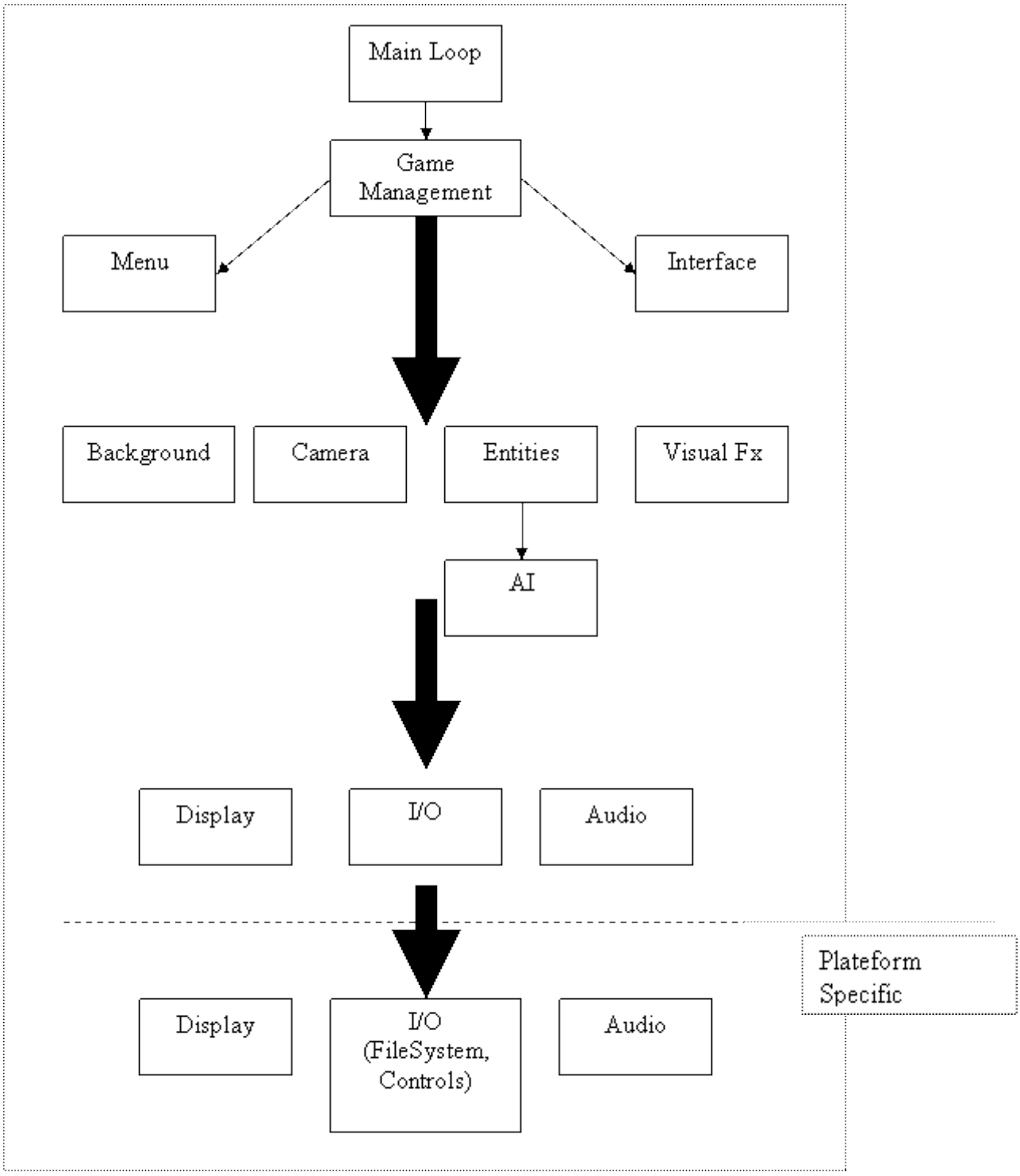
Techniques de Code

Path finding (calcul de l’itinéraire entre deux points)

L’univers est composé d’obstacles simple et convexes. Le path finding sera donc simple car les itinéraires ne peuvent pas être pris dans des voies sans issues. Entre deux objets, il y a toujours un chemin simple.

Annexe « Technical Design » - Partie Code

Schéma Game Architecture



ARCHITECTURE

Niveau 1 : (couches non dépendantes de la machine)
- Game Loop (gestion du jeu très haut niveau, appelle la mise a jour des managers en passant le delta time)

Niveau 2 :
- Game | Menu | Interface (Menu ou Interface, mais toujours jeu derrière). Prevoir aussi les mini jeux ici.

Niveau 3 :

- Entités (Gestion de toutes les entités. Créatures, papillons, décors, ballons, etc .... tout ce qui bouge). "Affichage". "Update". Créature gérée à partir des ordres donnés. Elle est quasiment complètement autonome.
- Background (Gestion du décor). "Affichage", Donne les infos pour les collisions.
- Camera (Gestion de la vue). Bouge la camera à partir des infos du joueur "Update". Découplé des créatures.
- Visual FX. Manageur d'effects speciaux (bulles, vapeur cascade, vapeur douche).

Niveau 4 : Couche basses, non dépendant de la plateforme.
- Display : Gestion d'objets 3D. Chaque objet a une position, une orientation et peut être animé (ajouter un gestionnaire d'animations). Chaque objet pointe vers un objet bas niveau.
- IO. Gestion des entrées sortie. C'est l'interface entre le haut niveau et le bas niveau. Donc pas grand-chose ici.
- Audio : Gestion des sons et musiques haut niveau. Interface avec le bas niveau.

Niveau 5 : Couche low level (dépendant de la plateforme)
- Display : Affiche des objects 3D / 2D. C'est ici que tout se fait. Les données sont chargées (au format de la plateforme). Fonctions : Load, Update, SetPosition, SetRotation
- IO : Gestion des entrées sorties. Fichiers, controles.
- Audio : Moteur son pour la plateforme.

OUTILS

- Exportation Objets Skinnés. (On travaille depuis MAYA, le format d'export est Collada.)

- Exportation des animations.

- Exportation Objets. (On travaille depuis MAYA, le format d'export est Collada. Il faut écrire un convertisseur Collada->G3o et Collada->PS2).
La première version montre que cela fonctionne bien, le système peut être utilisé. La version de collada utilisée est la 1.4.0.
L'export d'objets texturés se passe bien (reste à debugger sur PS2 le mapping qu'est un peu a coté).

- Création zones carte (PolyLigne ou plygone crée sous MAYA)
Idem que point précèdent. L'exporteur détecte le nombre de vertices dans un polygone et en déduis que c'est une polyline.

MOTEUR GENERIQUE

- Squelette de l'application (10 modules) (Voir graph architecture)

- Gestion Souris

- Animation persos Skinnés

Faire du code pour lire des anims et les enchaîner. Les anims n'ont pas de rapport avec le déplacement du personnage. Par contre les anims peuvent contenir un déplacement local du perso, comme par exemple un saut. L'anim représente le perso qui saute sur place.
Coder aussi l'enchaînement des animations, comment faire ça ? Faire ça sans transition ou faire un blend des animations pour avoir un truc sans coupure.

- Animations Objets

Le seul objet animé c'est la balançoire. Elle à un pivot et une simple rotation peu suffire. L'animation de la balançoire sera faite en code.

- Gestion Camera (Zoom, déplacements)

La gestion est simple, déplacement, zoom. Le travail consiste à gérer des transitions entre deux états.

- Audio sons. Audio musique

Intégration d'un moteur son haut niveau.
Et ensuite bas niveau (PC ok, PS2 multistream plus long).

- Affichage persos Skinnés

Ceci est fait par le moteur graphique en théorie donc pas trop de travail.

- Affichage Objets

Ceci est fait par le moteur graphique en théorie donc pas trop de travail.

- Localisation

Gestion d'un fichier multilingue. Faire un fichier Excel qui sera exporté au format texte (format CSV). La colonne a un id, ensuite on a la traduction langue par langue. On prévoit d'abord les langues les plus courantes (au niveau des pays les plus susceptibles de vendre notre jeu). Anglais, français, Allemand, Italien, Espagnol.
Pour le moment, pas de gestion de l'Unicode (pour des langues plus exotiques comme les pays de l'est ou l'asie). On fera sur demande.

- Module system : File system, mémoire et temps

Faire des petits modules haut niveau pour gérer tout ça avec une interface commune. Les modules bas niveaux existent déjà pour PC et PS2.

MOTEUR RONRONS

- Sauvegarde / Chargement (PC ok, PS2 bien plus long)

Sur PS2 il faut respecter toutes les règles imposées par les TRC (Technical Requirement Checklist) c'est donc beaucoup plus long (compter au moins 2 semaines).

- Déplacements créatures

Le déplacement est fait en code. Il faut jouer les animations en fonction de l'action.

- PathFinder (simple)

Faire un pathfinder local et un pathfinder global. Une créature sais rejoindre un point pas trop loin en reestimant son deplacement a chaque frame (ça donnera plus de vie). Créer une liste d'ordre pour atteindre une destination plus loin ou plus complexe (la map est faite pour qu'il n'y ait pas de destination complexe justement). Déplacement = Succession d'ordre « VA ICI ».

- Gestion zones écran

En fonction de la position de la souris, gérer les actions possible (c'est un system en plus de l'interface).

- Action Mange/boire
- Action Jouer

- Action Laver
- Action Faire ses besoins
- Action se faire gronder, féliciter …

Jouer une anim, et peut être une anim camera. Mettre a jour la créature. La créature passe dans la lecture d'un petit script en recevant cet ordre. Certaines actions déclenche des animations au hasard.

- Collision créature/sol

Le sol est une grille avec pas fixe et des altitudes. Donc la détection de la hauteur est simple.

- Collision créature/créature

Pas vraiment de collision, juste empêcher que les créature se rentre dedans, elle vont donc glisser les unes sur les autres. Chaque créature aura une enveloppe de collision en forme de cercle (soit une polyligne, soit un cercle).

- Collision créature/objets (ou zone)

Chaque objet a une forme de collision simple (polyligne ou cercle).

- Ia Créatures (15 actions) (Gestion d'ordre)

Les créatures utilisent des états ainsi qu'une liste d'ordre. Les ordres sont lus quand l'état le permet. Chaque ordre peut déclencher un petit script (codé) et l'état varie en fonction de ce script. Par exemple, la créature est dans l'état « attente », elle exécute le script attente (jouer tout seul), elle reçoit un ordre « aller a », elle termine le script en cours (termine l'anim en cours) et passe dans l'état « bouge » et exécute le script de déplacement. Chaque script a ses propres caractéristiques (peut être interrompus ou pas). Les ordre se cumulent ou s'annulent si ils sont contradictoires ou non compatibles.

- Gestion des jauges (5 jauges)

Chaque créature a en interne son état de santé (5 paramètres). Ces indicateurs changent suivant le temps ou les actions effectués.

- Etat Maladie

L'état maladie se traduit par une modification du comportement et des interfaces (voir game design). Les animations sont un peu différentes.

- Gestion des besoins créatures

Il faut gérer les besoins et les priorités de ces derniers. Voir game design.

- Interface Besoins créatures

L'interface est juste l'affichage. Au dessus de la tête de la créature un petit icône apparaît.

- Gestion Evolutions ronrons

Suivant le temps, les ronrons évoluent.

- Créature PNJ (5 créatures)

Elles obéissent à des ordres (tout comme les créatures joueurs). Les ordres sont générés par un script.

- Gestion Toboggan

Ceci est une entité (comme la créature et tout ce qui bouge ou peut être utilisé)

- Gestion Balancoire
Ceci est une entité (comme la créature et tout ce qui bouge ou peut être utilisé)

- Gestion Zone Rocheuse

Ceci est une entité (comme la créature et tout ce qui bouge ou peut être utilisé)

- Gestion Cascade

Ceci est une entité (comme la créature et tout ce qui bouge ou peut être utilisé)

- Gestion Anim scriptée papillons

Ceci est une entité (comme la créature et tout ce qui bouge ou peut être utilisé)

- Interface InGame

Simple affichage

- Interface Statistique

Simple affichage

- Curseur Dynamique

Suivant la position à l'écran, les actions sont différentes. Voir game design

- Gestion du temps

- Gestion Aide Dynamique
- Affichage Aide Dynamique

- Menus Mini jeux 3 pages

- Interfaces Mini jeux 6 pages

- Mini jeu Taupe

- Mini jeu marelle

- Mini jeu pêche

- Mini jeu tir ballon

- Mini jeu trouve cuicui

- Menu 6 écrans

Pour les polices de caractères on utilise un générateur de fonte bitmap depuis une fonte vectorielle. Sur www.dafont.com il y a tout ce qu'il faut comme fonte. Le logiciel de génération est « Bitmap Font Generator » sur angelcode.com.

- Customisation créature

Ceci est une option.

Règles :

- Commencer par les trucs les plus difficiles (exports, fontes)

- Faire d’abord la version PC et ensuite travailler sur un portage PS2

- Sur PC on utilise un écran de 640x512, c’est la résolution d’un téléviseur PAL. Ce sera plus facile pour les portages. Si on vend une vraie version PC, faudra prévoir une résolution configurable.

- Pour le jeu, faire peu de répertoires.