DOSSIER DE CONCEPTION TUNNELS AND TROLLS

SCENARIO :

Tunnels & Trolls va mettre le joueur dans la peau d'un personnage cherchant à faire fortune. Pour cela, il faudra piller une grotte infesté de monstre et aussi de trésors. Le plus gros trésor étant gardé par un puissant dragon.

PERSONNAGES :

Le joueur pourra choisir d'incarner 1 personnage parmi 4 types :

- L'archer
- Le magicien
- Le voleur
- Le guerrier

SYSTEME DE JEU :

Compétence des personnages :

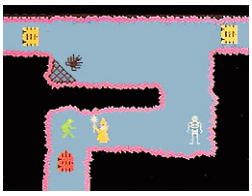
Chaque personnage possède une et une seule vie. n points de santé maximum, un certains nombre de « point de combat ». (munition, durée de vie de l'arme), un niveau de dégat et une vitesse.

- Au départ, l'archer possède un carquois avec 20 flèches. 2 Points de dégâts par coup. Vitesse = 2. 100 points de vie max.
- Le magicien à mémorisé 20 sorts d'attaques qu'il peut lancer. 3 Points de dégâts par coup. 70 points de vie Max. Vitesse=2.
- Le voleur possède 20 dagues qu'ils pourra lancer. 1 Point de dégât par coup. Vitesse = 3. 100 points de vie max.
- Le guerrier possède une épée qui pourra utiliser 40 fois avant qu'elle ne se casse (corps à corps). 120 points max. 3 Point de dégât par coup, Vitesse = 1 (lent)

Le labyrinthe :

Le héros devra trouver la salle du dragon dans un labyrinthe composé de 32*24 salles (768 salles). Les salles sont de 3 types :

- Labyrinthe



- Salle de garde / Salle de trésor / Cachot

Cette salle est obligatoirement gardée par un ou plusieurs troll et contient de l'or ou des bonus.

- Salle du dragon (1 seule et unique) : Dans cette salle contrairement aux autres salles, les bonus d'armes « respawn » régulièrement. Dans cette salle il y à forcément un dragon.

Le bestiaire qu'on peut tuer (8 maxi, contrainte technique):

Les monstres attaquent au corps à corps, et font perdre n point de vie par tour ou ils sont en contact avec le joueur.

- Trolls : Ces êtres répugnant se trouvent surtout dans des salles de garde. Il possèdent 3 points de vies, et font 2 points de dégâts par tour.
- Serpents : Ils sont lent, par groupe de 8, démarrent du centre de la pièce et vont chacun dans une direction et « rebondissent » sur les murs dès qu'ils en rencontre un. (1 point de vie, 1 point de dégât)
- Dragon : Boss final du jeu. 100 points de vie. Lance des boules de feu en direction du joueur qui font 20 points de dégâts. Si le joueur à trouvé la potion anti-flamme les dégâts seront limité à 5 points, à 0 points pour le chevalier uniquement.
- Dynamite : Objet à utiliser par le joueur. Si celui ci est trop prêt d'elle quand elle explose : 50 points de dégâts. (C'est pourquoi elle est considérée comme bestiaire).
- A compléter … (4 encore possibles)

Le bestaire qui respawn (8 maxi) :

- Squelettes : Contrairement aux autres monstres, une fois éliminés ceux ci se transforme en tas d'os. Si le joueur revient dans la salle/labyrinthe, ils seront de

nouveaux recomposés et prêt à attaquer le joueur. 2 point de vie, 1 point de dégât. Ils sont toujours par groupe de 2.

- Araignée rouge : Les araignées rouges ne bougent pas et n'attaque pas le joueur. Elles crachent des toiles d'araignées qui paralyse le joueur durant un certain laps de temps. (2 points de vie)
- Araignée verte : Les araignées vertes ne bougent pas mais attaquent le joueurs avec un salve de poison qui fait 10 points de dégâts. (2 Points de vie)

Les bonus (8 maxi contrainte technique):

- Recharge d'arme : Remet +5 aux armes des joueurs.
- Clé : Rouge
- Clé : Verte
- Anneau de force : Objet unique, double les dégâts
- Coffre : Augmente le nombre de pièces d'or trouvé
- Potion anti flamme : Objet unique pour le combat final
- Potion de soin partielle : + 20 points de vie.
- Dynamite

Les obstacles (8 maxi contrainte technique):

- Portes : Rouge, Vert → Nécessite une clé de couleur identique pour l'ouvrir. Une fois ouverte, la clé disparaît de l'inventaire.
- Porte Noire : Une clé, quelle quelle soit fait l'affaire.
- Rocher : Bloque une sortie. Nécessite de la dynamite pour passer.
- Trappes pièges : Invisible au joueur, elles s'activent quand le joueur passe dessus.
  - → Lancé de flèches (évitable)
  - → Trappe ouverte sur sol de pointe (1 points de dégâts par tour ou le joueur reste dessus)

- A compléter … (2 encore ...)

Combat Final :

Le combat final à lieu dans l'antre du Dragon. Celui-ci se déplace latéralement et tire des boules de feux sur le héros.

En dessous du dragon 4 ou 5 rochers derrières lesquelles ont pourra se cacher pour éviter les tirs. Et régulièrement les armes seront respawneés. Le combat devrait être plus accessible pour les personnages qui peuvent tirer à distance.

La potion anti flamme devra être obligatoire pour le chevalier pour ne pas mourir.

CONCEPTION TECHNIQUE :

Ce chapitre propose un début de réflexion sur la conception technique.
Le labyrinthe sera codé en ROM. Les portions de labyrinthe seront des « plaques » précalculés (soit stocké en rom, soit créé par programmation).

Les salles sont numérotés ainsi :

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,...

La structure globale du labyrinthe sera ainsi en mémoire :

SALLE 0 : Type de plaque (Plaque de Labyrinthe N°,Salle de garde ,Salle du dragon),
Monstres présents non respawn (1 octet),
Monstres présents respawn (1 octet),
Bonus présents (1 octet),
Obstacles présents (1 octet),
SALLE 1 :
SALLE 2 : etc etc

Taille en mémoire : 3840 octets

Le mode vidéo utilisé sera le MODE 2 TEXT

Ce tableau en ROM, sera recopié dans la VRAM à 3 adresses différentes.
1 plage pour les monstres présents (qui ne respawne pas !!).
1 plage pour les bonus présents
1 plage pour les obstacles présents

En effet, si on à ouvert par exemple une porte dans une salle, il ne faut plus que cette porte soit de nouveau refermée quand on y revient.

C'est le programme qui déterminera, selon la plaque à afficher , ou mettre les bonus/monstres/trésors.

HEROS, MONSTRES ET OBJETS :

Les héros et monstres seront toujours vu de coté (direction droite ou gauche), et occuperont 2 sprites monochromes (16*32). (Voir les mouvement de Dungeon and Dragon Intellivision)
Certains objets seront dessinés en caractères (portes, rochers, trappes), d'autres en sprites de 16*16 (potions, bonus d'armes etc etc …).

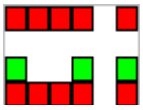
ZONE DE JEU, PLAQUES et TILES:

La zone de jeu, dans laquelle sera affichée le décor est de 30*20 caractères. Les 4 autres caractères restant en Y seront pour afficher la vie, l'or, et les objets dans l'inventaire.

Pour donner une certaines variété au labyrinthe il va falloir stocker en mémoire un certains nombres de plaques labyrinthe. Une plage sera composée de tiles. Chaque tiles faisant 40*40 pixels (5*5caractères) , une plaque sera donc composée de 6*4 tiles.

Pour gagner en mémoire, 1 plaque sera codée du 4 octets.

Exemple d'une plaque :



Sera encode ainsi :

Octet 1 : 111101 11
Octet 2 : 000000 00
Octet 3 : 100101 01
Octet 4: 111101 11

Les 2 derniers bits de chaques octets donne le type de tiles graphique à utiliser pour le mur (bit à 1 des 6 autres bits).

Dans cette plaque, il faut pouvoir positionner les monstres, les objets etc etc …
Considérons la plaque composée de tile ainsi :

t0,t1,t2,t3,t4,t5
t6,t7,[...]

On met en place une table des objets autorisés sur cette plaque et de leur position. Une table par type d'objet (NoRespawn,Respawn,...)

NoRespawn :
Araignée Rouge, t13
Serpent,t10

Respawn :
Potion,t16

Sur cette plaque seul les araignées et les serpents et une potion est affichage respectivement aux position 10,13,16.

Donc pour cette plaque, en mémoire, coutera 4 + 6 octets = 10 octets.

PSEUDO ALGO :

Variables :

```
const byte gfxTileMurs[] = {c1,c2,c3,c4,[...]   // 25 octets pour une tile
};

const byte plaques[] = {11110111,00000000,10010101,11110111, // Plaque type 0
                        11110111,00000000,10010101,11110111, // Plaque type 1
                        11110111,00000000,10010101,11110111}; // Plaque type 2

const char autorisePlaqueNoRespawn0[] = {ARAIGNE,10,-1
}

const char autorisePlaque1[] = {TROLL,11,-1
}

const byte autoriseGlobalNoRespawn[] = {*autorisePlaque0,*autorisePlaque1};

byte LabyGlobal[] = {
        Type de plaque
        Monstres présents non respawn (1 octet),
        Monstres présents respawn (1 octet),
        Bonus présents (1 octet),
        Obstacles présents (1 octet),
};

void copieLaby2Vram()
{
    disable_nmi();
    screen_off();
    // Copie les informations des monstres non respawn, bonus et obstacle en VRAM
    screen_on();
    enable_nmi();
}

void GénèrePlaque(int zoneLabyNo)
{
    int index;
    byte typePlaque;
    byte montresNoResp;
    byte monstresResp;
    byte Bonus;
    byte Obs;

    index = zoneLabyNo*5;
    typePlaque = (* zoneLabyNo+index); // N° de la plaque
    monstresResp = (* zoneLabyNo+index+2);

    monstresNoResp = AdresseVram+plaqueNo;
    Bonus = AdresseVram+plaqueNo;
```

```
	Obs = AdresseVram+plaqueNo;

	// On dessine la plaque
	for (y=0;y<4;y++)
	{
		tmp = 128;
		type_de_mur = (*plaque+typePlaque*4+y) && 3;
		for (x=0;x<6;x++)
		{
			mur =  (*plaque+typePlaque*4+y) & tmp;
			if (mur==1)
			{
				put_frame0(x*5,y*5, *(gfxTileMurs+type_de_mur*25),5,5);
			}
			tmp = tmp >> 1;
		}
	}


	// On crée les monstres non respawn
	tmp = 128;
	sortie = 0;
	while ( sortie==0)
	{
		if (tmp==1) sortie = 1;
		 pointeur = autoriseGlobalNoRespawn[typePlaque];
		 if (monstresNoResp&tmp) = tmp
		{
			while(*pointeur!=-1)
			{
				if (*pointeur==tmp)  // Le troll à le code 128 par exemple dans le
tableau
				{
					ptroll = (*pointeur+1);
					// Crée le troll en position x,y par rapport au n° de la tile
récupéré
					break; // On quitte la boucle
				}
				*pointeur+=2;
			}
		}
		tmp=tmp/2;
	}

	// On fait la même chose pour les autres types d'objets

}
```