**Erwan Palanque**

0837003

**Projet Synthèse**

420-B65-IN

Présenté à

**Jean-Christophe Demers**

Vendredi 21 mars 2013

Cégep du Vieux-Montréal

Contenu du document

Devis.......................................................................................................................................................1

Description précise du projet...................................................................................................................

* Spécifications techniques…………………………………………………………………1
  + Langages
  + Système de sauvegarde de la progression
  + Implémentation du contrôle du temps
* Histoire…………………………………………………………………………………...1
* Description du gameplay…………………………………………………………………2

Diagramme UML...................................................................................................................................3

Cas d’usages....................................................................................................................................4-5-6

Devis

Ce projet synthèse sera un jeu de type plate-forme. Le joueur contrôlera un personnage animé et devra résoudre des énigmes pour finir l'histoire. La particularité du jeu sera que le joueur aura le moyen de faire un retour dans le temps, et influencer ainsi ses actions futures et passées ainsi que celles de ses ennemis. Il devra aussi se servir du retour dans le temps afin de résoudre des énigmes de plus en plus complexes.

-------------------------------------------------------------------------------------

Description précise du projet

**Spécifications techniques:**

- Langages :

Python, PyGame

- Système de sauvegarde de la progression :

Les éléments importants (vie, liste d’objets, points, etc.) liés au joueur, ainsi que le niveau dans lequel se trouve le joueur, seront sauvegardés dans un fichier externe.

*- Implémentation du contrôle du temps :*

Plusieurs techniques sont envisageables, la plus probable me semblant être un « Command Pattern ». De cette façon, toutes les actions effectuées au cours du jeu seront sauvegardés, au passage d’une des fonctions traitante, dans une pile à taille fixe. Lorsque le joueur actionnera la commande de retour dans le temps, le flot d’action régulier (trame principale) stoppera, et la pile générer se dépilera et enverra toute les actions, inverses de celles sauvegardés, au contrôleur, afin de former une seconde trame. Tant que le joueur le souhaitera ou jusqu’à ce que la pile soit vide, tous les écouteurs seront muets et la seconde trame défilera. Si la pile est vide, la trame régulière reprend et les écouteurs sont réactivés.

**Histoire :**

Le personnage principal se nomme Fred. Fred est emprisonné dans un bâtiment spatial de haute sécurité dont aucun prisonnier ne s'est jamais échappé. Les seuls êtres biologiques des bâtiments sont les prisonniers. La prison est gardée par des robots, des drones, qui déclencheront des pièges ou attaqueront s'ils découvrent un prisonnier hors de sa cellule. Elle est aussi gardée par des caméras de sécurité, des lasers, et d’autres zones de détection en tout genre. Un champ de force spatial fait en sorte d'attirer tout vaisseau se trouvant à proximité de la station s'il est activé. Fred va tenter de s'enfuir, mais avec un avantage sur tous ses prédécesseurs : il peut contrôler le temps sur une courte période. Il va donc essayer de déjouer les pièges et le système informatique installés dans la prison pour atteindre le spatioport et voler un vaisseau spatial afin de s’évader !

**Description du gameplay :**

Le jeu offrira deux difficultés différentes : normal et difficile. Avec le mode difficile, le joueur n’aura qu’une seule vie pour finir le jeu, et lors d’un échec, il devra recommencer du début. Le mode normal lui laissera 3 vies, et un échec le ramènera au début de l’étape actuel.

Il y aura 3 grands niveaux à traverser:

- La première sera de trouver un moyen de sortir de sa cellule et de trouver un plan de la station, afin de se repérer plus facilement dans ce bâtiment gigantesque, car lorsque le système de sécurité découvrira que Fred s'est échappé, il activera ses défenses, et un compte à rebours sera enclenché. A la fin du temps impartie, la zone concernée de la prison sera privée d’oxygène Il ne restera alors à Fred que 30 secondes pour évacuer les lieux vers la prochaine zone, le temps que le processus de vidage soit complété. Le joueur n'est pas obligé de chercher une carte, mais il sera alors plus long pour lui de trouver une issue vers le prochain niveau, et ses chances de survie face au compte à rebours s'amenuiseront.

- Le deuxième niveau consiste à désactiver le champ de force, le bouclier et le système de défense externe de la station spatiale, sans quoi Fred ne pourra jamais décoller et s'évader. Les salles de contrôle sont séparés les unes des autres par différents systèmes de sécurités. Fred devra déjouer, contourner et se servir de son pouvoir de contrôle du temps pour atteindre les salles de contrôle. Certains objets seront invariables au temps, c'est-à-dire que le retour dans le temps de Fred ne les affecteras pas, ce qui sera nécessaire pour réussir à passer certaines parties de la station. Cette section aura également un compte à rebours au niveau de l’oxygène, qui sera enclenché soit par un drone, soit lorsque le premier système sera désactivé par le joueur. Une fois encore, lorsque le compte à rebours sera terminé, Fred aura 30 secondes pour s’échapper vers l’étape suivante.

- Le troisième et dernier niveau consiste à trouver le hangar et réussir à décoller. Le boss final sera présent dans le hangar.

Diagramme UML



Cas d’usage

**Écran d’accueil**

* *Nouvelle partie :*
  + Changement de « canevas » 🡪 vers NouveauJoueur
    - Le joueur entre son nom et appui sur Entrer
      * Initialisation des variables pour une nouvelle partie
      * Changement de « canevas » 🡪 vers Niveau1
        + Vers le cas d’usage **Jeu**
* *Charger une partie :*
  + Changement de « canevas » 🡪 vers chargerPartie
    - Ouverture du fichier de sauvegarde et afficher chaque partie disponible
    - Le joueur choisit une partie
      * Entrer
        + Chargement en mémoire des variables de la partie sélectionné
        + Chargement du niveau actuel de la sauvegarde sélectionné

Vers le cas d’usage **Jeu**

* + - * Supprimer
        + Suppression de la sauvegarde
    - Annuler
      * Changement de « canevas » 🡪 vers Splashscreen
      * Retour à l’écran d’accueil
* Quitter
  + Ferme le jeu

**Jeu**

* Génération des objets et ennemis
* Trame principale lancée. Chaque classe effectue sa prochaineAction() : tous les robots et pièges vérifient la position du joueur par rapport à la leur (influencé par la portée de détection de chacun). Les drones bougent de gauches à droites.

*Joueur* :

* Vers le haut, le bas, la gauche ou la droite :
  + Vérification chemin dans direction correspondante
    - Si chemin libre :
      * Changement de direction
* Retour dans le temps :
  + Vérification de la pile
    - Si non nulle :
      * Timer trame principale en pause
      * Dépilage de la trame secondaire à la même vitesse qu’à l’empilage
      * Si pile vide ou si le joueur arrête le retour dans le temps :
        + Redémarrage du timer de la trame principale
    - Sinon
      * Afficher message d’avertissement
* Interagir avec un objet :
  + Vérification de la proximité de l’objet
    - Si suffisamment proche :
      * Changer l’état de l’objet
        + Si l’objet est un interrupteur de piège :

Supprimer le piège correspondant de la liste des pièges

* Ouvrir la carte
  + Affichage du « canevas » de la carte par-dessus le canevas actuel
  + Affichage d’un point indiquant la salle dans laquelle se trouve le joueur par rapport au plan de la station.
  + Une seule option : Sortir
    - Effacer le « canevas » de la carte
* Se faire attaquer
  + Si en mode normal :
    - S’il ne reste qu’une vie :
      * Si la pile de la trame secondaire n’est pas vide :
        + Le joueur à 3 secondes pour l’activer

Si non-activé, le joueur a perdu

* + - Sinon :
      * Perd une vie
      * Devient intouchable pendant 2 secondes
  + Si en mode Difficile :
    - Si la pile de la trame secondaire n’est pas vide :
      * Le joueur à 3 secondes pour l’activer
        + Si non-activé, le joueur a perdu
* Attaquer
  + Parcours de la liste d’ennemi
  + Si ennemi suffisamment proche :
    - Attaque
* Changement de niveau
  + Sauvegarde des variables et du niveau dans le fichier externe

*Robots*

* Si joueur détecté dans la zone :
  + Si Drone de Surveillance :
    - Le drone devient immobile
    - Des pièges sont rajoutés dans la salle
  + Si Drone de Combat :
    - Le drone avance vers Fred
      * Si le drone est suffisamment proche de Fred :
        + Attaque Fred
  + Si Caméra :
    - Des pièges sont rajoutés dans la salle

*Pièges*

* Si piège détecte Fred à proximité
  + Attaque
  + Si détecteur :
    - Le piège est retiré de la liste des pièges

*Boss*

* Se déplace de gauche à droite, de haut en bas
* Toutes les 7-8 secondes :
  + Attaque
* Si le niveau de vie est à 50% :
  + Vitesse augmenté
  + Délai réduit entre les attaques
* Si le niveau de vie est à 0 :
  + Changement de « canevas » :
    - Fin du jeu