1. **CtrlPad.as** : Contient les instructions et fonctions reliées aux déplacements. Elle s’assure aussi des déplacements diagonaux. Le tout est fait en calculant l’angle du personnage sur la scène.
   1. loop() : Le loop exécute les instructions à chaque frame. Il questionne l’angle du personnage puis détermine ainsi sa direction. Il renvoie ensuite le bon frame d’animation et initie le déplacement selon la direction.
2. **Debogage.as :** Offre des commandes de debogage accessible avec des series de touches. Performe différentes actions comme ajouter un compagnon, déclencher un bataille, ajouter un patate, etc.
   1. frappeClavierDebogage() : Écoute les touches appuyées et actionne le mode debogage.
   2. verifierMDP() : Demande d’entrer un mot de passe avant d’activer le mode debogage. Elle va comparer la chaine entrée et celle attendue et va active ou non le mode.
3. **Dialogues.as :** Cette classe s’occupe de l’apparition des dialogues.
   1. declencherEtape() : Cette function va afficher le nom du personage qui parle, suivi de sa réplique tirée du tableau de PNJ.as. En plus, il gère la durée et l’effacement des dialogues.
   2. quitterDialogue() : Garde en mémoire l’heure à laquelle le joueur à déclenché ce dialogue pour ne pas le ravoir immédiatement.
4. **Etre.as :** C’est la classe qui s’occupe des interactions des personnages. Les points de vie, l’attaque, les animations, les statistiques, etc.
   1. Blesser() : C’est l’attaque. Elle récupère un nombre, le damage, et va le soustraire à la vie de l’attaqué. Si l’attaqué n’a plus de points de vie, afficher l’animation avec le label Mort.
   2. Guerir() : Par défaut, les personnages ne peuvent pas ressusciter. Mais, s’il le peuvent, la function reçoit un nombre et ramène le personage à la vie avec ce nombre de points de vie.
   3. jouerAnim() : La function va recevoir un label d’animation. Si ce label ne correspond pas à l’animation qui joue présentement, il va verifier si l’animation demandée existe. Si oui, il va la jouer. Sinon, il va averter l’utilisateur qu’elle n’existe pas.
5. **Jeu.as :** Gère différents aspects de la partie, comme l’affichage des niveaux et pages-écrans, les dialogues, les clips, etc.
   1. Jeu() : Initialisation des statistiques de depart des personnages.
   2. changerEcranJeu() : Gère l’affichage des différents niveaux et pages-écrans.
6. **Log.as :** Cette function permet d’afficher facilement un message à la manière d’un trace.
   1. Log() : On lui donne une chaine (?) et une priorité. Ces informations seront ensuite envoyées dans le trace.
7. **MenuInfo.as :** C’est l’affichage du menu d’information, qui contient les statistiques, les compagnons, les objets de l’inventaire, etc.
   1. afficherLesObjets(): Affiche la liste des objets que le joueur possède. S’il ne possède rien, il affiche (Aucun objet).
   2. frappeClavierMenuInfo() : Gère les touches ENTER et I, qui affichent et quittent le menu.
8. **Monstre.as :** La gestion des monstres est faite ici, comme les statistiques et l’état.
   1. initParam() : Initialise les paramètres de depart pour les différents attributs, comme les points de vie, la defense, etc.
9. **Objet.as :** Gestion des éléments interactifs, comme les patates, l’ori, les barrières, etc
   1. Interagir() : À la rencontre d’un objet, crée la réplqie qui sera montrée au joueur. Exemple: Vous avez trouvé une patate.
   2. verifierSiValide() : Vérifie si l’objet rencontré ou une pièce d’or, une patate ou une barrière.
10. **Obstacle.as :** Elle s’occupe des collisions et de l’opacité des zones de collision.
    1. gererVisibiliteZone() : Si getZonesTechniquesVisibles est true, les zones seront visibles pour le debogage. Sinon, l’opacité des zones de collision sera mise à 0.
11. **Perso.as :** La gestion du personage est faite ici, comme les statistiques et l’état.
    1. initParam() : Initialise les paramètres de depart pour les différents attributs, comme les points de vie, la defense, etc.
    2. superForceDebogage() : Si cette commande de debogage est activée, les statistiques seront multipliées par 10.
12. **PNJ.as :** C’est la gestion des interactions avec les PNJ.
    1. Interagir() : Contient la liste de tous les dialogues possibles pour chaque PNJ interactif. La fonction choisit une phrase parmi le tableau tSequencePossibles.
    2. verifierSiValide() : Vérifie si le clip est un PNJ interactif en vérifiant ses attributs comme son nom et s’il a un dialogue.
13. **Porte.as :** Gestion des portes (zones de téléportation).
    1. gererVisibilite() : Si getZonesTechniquesVisibles est true, les portes seront visibles pour le debogage. Sinon, l’opacité des portes sera mise à 0.
    2. Interagir() : Quand le joueur demande une teleportation, cette function l’appelle par son nom. Si elle ne la trouve pas, elle retourne Erreur.
14. **Tableau.as** : Elle s’occupe d’amener le joueur dans les différentes zones accessibles par les portes et les zones de transportation.
    1. initParam() : cette function va recueillir la position voule du joueur à l’arrivé. Il ajoute ensuite le joueur à cette position.
    2. placerSandwichPerso() : Cette function détecte si le joueur devrait être en dessous ou au dessus d’un autre clip, et execute les arrangements nécéssaires.
    3. deplacerJoueur() : C’est la gestion de la direction du joueur, dans le cas où la fonction regarde si le joueur sera arrêté (par un obstacle ou un dialogue) au prochain déplacement.
15. **Collision.as :** Si le parent est un objet de type Obstacle, cet objet obtient une zone de collision. (une seule fonction)
16. **Combat.as :** C’est la gestion des combats comme l’ajout des clips des monstres et des joueurs, ainsi que l’initialisation de leurs paramètres. En plus, c’est la gestion des touches et des actions, des tours et de victoire/mort.
    1. Victoire() : Cette function s’occupe de l’affichage des gains comme l’XP et l’or, en plus d’enlever les clips des monstres si le joueur gagne le combat.
    2. creerChef() : Cette function s’occupe de l’initialisation de Torgul; ses statistiques et son MovieClip.
    3. creerMonstres() : S’occupe de faire apparaitre des monstres, dont le nombre depend d’un random. La function s’occupe de l’initialisation des clips et des statistiques.
    4. frappeClavierCombat() : S’occupe des touches et des actions qui y sont associées lors d’un combat, comme Attaque, Magie ou Fuite.