CONCEPTION

Classes :

Actor : on a decide d'introduire un attribut protected final PLAYER_PRIORITY qui est disponible a tous les autres acteurs afin de reconnaitre que l'acteur avec lequel ils interagissent est un player (tout cela en évitant un test de type « instanceof »)

Overlay(hérite de Actor) Nous avons modifie la méthode draw ou la boucle for va désormais jusque healthMax et non la valeur brute 5.

Arrow(hérite d'Actor): flèche lancée en ligne droite(pas d'influence de gravite) par un ArrowBox ,inflige un dégât FIRE de 1 au Player qu'elle touche et disparait.

End (herite de Actor): un Acteur qui rend la transition entre les niveaux plus subtile.Il a 2 attributs TIME final initialise a 1 et duration incrémenté par deltaTime dans update qui change la transparence de la transition en fonction de l'écart entre TIME et duration et passe a un autre niveau des que duration surpasse TIME.

Background(hérite de Actor): Acteur qui dessine un arrière plan non solide.On lui assigne un PRIORITY dans son constructeur afin de choisir si cet acteur se dessine sur d'autres acteur ou non.

TurningSpike(herite de Actor): Un obstacle statique qui est en rotation autour de lui meme et inflige un dégât FIRE de 1 a l'Acteur en contacte avec lui. Pose un cooldown de 0.5 pour limiter les dégâts causes.

LevelDoor(herite de Actor): Acteur qui détecte si une souris est dans la box et clique pour renvoyer vers un level pris en attribut et initialise dans le constructeur.

Particle(herite de Actor) : Acteur qui représente une image aparaissant pendant quelques secondes puis disparaissant.Elle contient des setters pour tout ses attributs.

Floor(herite d'Actor): Acteur simple solide qui cree des blocks ensemble pour créer une plateforme qui prend le vecteur minimal (coin bas , gauche) et le vecteur maximal (coin haut,droit) de la plateforme.Qui prend une String et le concatene avec .left,.right et .middle pour changer les tiles a l'aide d'une boucle for.

FlyingProjectile(hérite d'Actor) :Projectile propulse vers une vitesse initiale donnée par un FlyingEnemy. Cause un dégât FIRE 1 au Player avec lequel il entre en contact et cree une particule smoke. Possede un cooldown pour éviter qu'il essaie de se desenregistrer 2 fois lors d'un contact avec de 2 objets solide simultanément.

FlyingEnemy(hérite d'Actor) : Acteur ennemi lance FlyingProjectile volant si le Player entre dans le champ de tir et cause des dégâts. Possède un attribut cooldown decremente par deltaTime dans update et qui est reinitialise a 1 après que le projectile soit lance. Possède un attribut boolean dead qui devient true si son health passe a 0 , en appelant la méthode die(). Sa méthode update régit le mouvement de FlyingEnemy a l'aide d'une fonction sin(delta) tel que delta est incrémenté, applique sur la vitesse pour qu'il fasse des aller -retours dans un secteur de notre choix. Il est

dessine en fonction de sa vitesse(vitesse>0-mouche vers la droite , vitesse<0-mouche vers la gauche).

SpaceInvader(herite FlyingEnemy): Un FlyingEnemy dont la méthode draw le dessine avec une sprite bien precise.

GroundEnemy(hérite d'actor): Un ennemi qui se déplace en oscillant autour d'une position.Son mouvement est conditionne par la méthode update qui utilise la fonction sin(delta) et sin(4*delta) associes au composante de la vitesse pour obtenir plusieurs oscillations. Il dessine une scie tournante sur elle meme et qui se déplace. Il contient les plusieurs getters et setters pour permettre l'access au attribut a ses classes héritantes comme Frog.

Frog (hérite de GroundEnemy): Un ennemi qui saute devant et derriere et qui inflige un dégât FIRE au Player a son contact. On surcharge la méthode draw pour pouvoir le dessiner allant vers la droite et vers la gauche dans différents états.

Slime(herite de GroundEnemy) : Un GroundEnemy qui prend en attribut supplémentaire un boolean pour choisir si lors de sa mort il laisse place a deux slimes plus petits ou non. Ceci est implemente au seins de sa méthode die(). La méthode update code le déplacement sinusoïdale de Slime. La méthode draw dessine le Slime vers la droite puis vers la gauche dépendant de sa direction(velocity.getX()< ou>0)

ArrowBlock(hérite de block) : block solide qui lance une flèche quand un Player est dans son champ de tir(devant lui et dans un intervalle de 2 sur les Y) avec un cooldown de 3.

Destructible(herite de Block): un block qui peut être détruit .Il possède 2 points de vie et son image se modifie en fonctions des dégâts reçus. Retourne ses points de vie actuels a l'aide d'une méthode getHealth().

HeartBlock(herite de Destructible): Un block destructible qui laisse place a un Heart lors de sa destruction ,au seins de sa méthode update.

PlaySound: utilisation de la classe pour utiliser du son au format mp3 au lieu de WAV. importation de la librairie javafx (sur Eclipse e(fx)clipse).

Nous nous sommes inspires de ce post http:// stackoverflow.com/questions/ 22490064/how-to-use-javafx- mediaplayer-correctly pour coder cette classe.

Cette classe appelle dans son constructeur un string qui constitue le path vers un media. ce media est ensuite ajoute au mediaplayer qui va jouer cette chanson en boucle a laide de la constante MediaPlayer.INDEFINITE. Nous avons aussi cree une méthode onEnd() qui est appelé par program après display.close et qui s 'occupe d'arrêter le son et de libérer les ressources associes au mediaplayer.

On utilise System.exit(0) dans la méthode onEnd() pour empêcher Java(TM) Platform SE Binary de rester dans le processus en cours de la machine(méthode proposée dans stackoverflow.com)

source : http://www.eclipse.org/efxclipse/index.html

Dans le cas ou cette classe pose problème lors de la correction (a cause de la librairie importée) nous avons aussi a disposition une archive ne contenant pas cette classe et qui n ‘y fait pas appel dans program.Vous trouverez dans ce lien la version du projet sans son : https://www.dropbox.com/s/eqfswjt47a7057v/PlatformGameNoSound.zip?dl=0

Xor (implemente Signal) : prends 2 signaux en attribut pour créer un Signal conditionne par la porte logique Xor.

Constant (implemente Signal): renvoie un signal constant qui depend de la valeur booléenne donnée au constructeur.

Licences :
Les assets ajoutées on été prise de www.kenney.nl

Music : http://opengameart.org/content/platformer-game-music- pack of the user CodeManu.

———————————————