Fichier CONCEPTION

Aspect Visuel

i. Ajout d'un effet de transparence sur certains acteurs :
   - la transparence est appliquée dans la méthode draw() de l'acteur avec la méthode void drawsprite(Sprite sprite, Box location, double angle, double transparency). Le double transparency est choisi selon l'acteur et l'état dans lequel il est.
   - Exemples :
     - les Portes, lorsqu'elles sont ouvertes (=signal associé est actif) : transparency de 0.25, sinon 1 (lorsqu'elles sont fermées). « output.drawSprite(getWorld()...., getBox(), 0, (closed ? 1 : 0.25)); »
     - les Coeurs, lorsqu'ils sont pris : transparency de 0.2, sinon 1.

ii. Ajout d'un effet de « flottement » sur l'acteur Clé :
   - ajout d'un attribut scl (=scale) double qui varie entre -0.5 et 0.5 dans la méthode update() : scl=scl ± input.getDeltaTime();
   - le signe de l'affectation d'au dessus est déterminé par un attribut up booléen ajouté également. La valeur de vérité de l'attribut up dépend de la valeur de scl. Si scl atteint la valeur -0.5, l'acteur est « au plus bas » de sa position, up devient true et l'acteur change de direction de mouvement et monte dans les airs, le contraire se produit lorsque scl atteint 0.5 . Les deux attributs scl et up sont initialisés dans le constructeur à 0 et false respectivement.
   - Le tout est géré est dans la méthode getBox() { return new Box(position.add(new Vector(0, 0.3).mul(scl)), SIZE_KEYS, SIZE KEYS); } qui influe sur la méthode draw() et donc sur le dessin de l'acteur Clé.

iii. Globalement, les tailles ainsi que les sprites des différents acteurs ont été modifiées. (Tous les sprites proviennent du Platformer Art Complete Pack de http://opengameart.org/content/platformer-art-deluxe)

Aspect « Gameplay »

i. Ajout d'un « temps de vie » à l'acteur Boule de feu (temps au bout duquel, la boule de feu disparaît). Ceci empêche la « sur-population » de boules de feu lorsque le Player lance beaucoup de ces dernières.
   - ajout d'un attribut lifeTime de type double initialisé dans le constructeur.
   - la valeur de lifeTime diminue selon lifeTime-=input.getDeltaTime; dans la méthode update().
   - Lorsque lifeTime atteint 0, l'acteur se désenregistre du monde dans la méthode update().

ii. Ajout d'une classe Plate-forme qui permet de créer des platformes de largeur 0.55 (limitation du sprite, la largeur doit être fixée pour toutes les plate-formes) et de longueur choisie (attribut width) selon une position (Vector position en attribut). Ceci permet la construction de plate-formes plus fines que les Blocks habituelles.
   - Les attributs with et position sont initialisés dans le constructeur.
   - La méthode getBox() définit la zone de collision à partir du vecteur position et de width. Le centre est de la plate-forme est situé : position.add((0.5*(width-1), 0.20)) - l'ajout de la composante y à 0.20 est due au dessin du sprite. Le sprite est dessiné tout en haut de l'image et non pas au centre.
   - La méthode draw() dessine la plate-forme avec une boucle For qui itère une variable i de 0 à width et selon la valeur de i, la méthode applique le bon sprite :
   - for (int i = 0; i < width; i++) {
     if (i == 0) sprite = "chocoHalfLeft";
     else if (i == width - 1) sprite = "chocoHalfRight";
     else sprite = "chocoHalfMid";
     output.drawSprite(getWorld().getLoader().getSprite(sprite), new Box(position.add(new Vector(i, 0)), 1, 1));
     }
   - Remarque : présence d'un attribut signal (initialisé dans le constructeur) permettant l'apparition et la disparition des plate-formes selon la valeur de vérité du signal.

iii. Modification des aspects de Player :
   - Lorsqu'une touche de direction est appuyée, la vélocité du Player augmente jusqu'à une certaine valeur, ceci est la conception proposée. L'ajout consiste en l'implémentation dans la méthode update() de Player, du fait que lorsqu'une des touches (droite ou gauche) est relâchée, la composante verticale de la vélocité devient nulle. Ceci permet une meilleure dynamique et un meilleur contrôle de son personnage.
   - Surpression de l'attribut booléen (colliding) et remplacement par deux attributs booléen (onGround) et (againstWall) permettant une meilleur compréhensibilité et une meilleur gestion des cas de collisions.
   - Suppression de la vélocité en paramètre du constructeur. La vélocité est maintenant directement initialisé au Vector.ZERO .

iv. Ajout d'un attribut cooldown de type double au Levier
   - Initialisé dans le constructeur.
   - La valeur de cooldown diminue dans la méthode update() si le levier est activé et est réinitialisé à chaque « activation » du levier. Lorsque cooldown devient négatif, le levier se « désactive » (= son attribut on devient false)

v. Modification des dégâts infligés par Spike
   - La conception originel propose que les Spikes appliquent des dégâts de type PHYSICAL. Je trouvais mieux de remplacer le type PHYSICAL par VOID, ainsi les Spike tuent instantanément les acteurs (plus spécifiquement le Player)

vi. Ajout d'une classe TileMap
   - Similairement à la classe Plate-forme, la classe TileMap permet de créer des grilles de taille mxn, composés de Blocks de taille 1x1 afin d'éviter de créer « manuellement » chaque Block.
   - La classe possède trois attributs initialisés dans son constructeur (Vecteur position, un double width et un double heigth)
   - la méthode getBox() créer la zone de collision, similairement à la classe Plate-forme mais sur les deux dimensions, càd : new Box(position.add(new Vector(0.5 * (width - 1), 0)).sub(new Vector(0, 0.5 * (height - 1))), width, height);
   - Et la méthode draw() dessinent selon une double boucle For, les Block de haut en bas, puis de gauche à droite et appliquent le bon sprite au bon endroit.

- for (int i = 0; i < width; i++) {
  for (int j = 0; j < height; j++) {
  if (j == 0) sprite = "choco";
  else if (i == 0 && j == height - 1) sprite = "choco.BottomLeft";
  else if (i == width - 1 && j == height - 1) sprite = "choco.BottomRight";
  else sprite = "choco.Center";

  output.drawSprite(getWorld().getLoader().getSprite(sprite), new Box(position.add(new Vector(i, 0).sub(new Vector(0, j))), 1, 1));
  }
  }
- Par Exemple : on veut que le coin inférieur droit du Block inférieur droit soit rogné, alors if (i == width - 1 && j == height - 1) sprite = "choco.BottomRight";

vii. Ajout d'une énumération Colors pour les couleurs des clés et des portes