Guyana Dream - Initialisation

Structure du jeu

– Préparer les dossiers et les fichiers dans le dossier du jeu
– dossier jeu
  jeu.html
  – sprites
    choix-joueur1.png
    choix-joueur2.png
    choix-pnj1.png
    spritesheet-joueur1.png
    spritesheet-joueur2.png
    spritesheet-pnj1.png
  – tilesets
    tileset.png
    fruit1.png
    fruit2.png
    fruit3.png
  – js
    carte.js
    persos.js
    evenements.js
    excanvas.js
  – json
  – css
    style.css
– Préparer la structure de la page HTML

Fichiers js

– Définir les variables globales
  – taille du canvas
– Instancier le jeu
  – Instancier les différents canvas (par leur z-index en particulier)
  – Lancer l'écoute des événements
  – Instancier les différents objets en appelant leur classe
    – La map et les persos héritent d'imageTuile. Pour les peros, celle-ci doit inclure les éléments d'imageDiv
    – La map appelle le tileset et la tuile et s'affiche sur certains canvas – Les tableaux sont placés dans un fichier json, qui doit être appelé par une méthode
    – Les persos appellent la spritesheet
    – Les objets interactifs appellent les événements
    – Les inventaires sont reliés aux objets interactifs et aux événements
    – Les score (et points) aussi
  – Appeler les méthodes liées aux objets
  – Appeler les méthodes liées aux objets interactifs quand un événement survient.

Les classes

Jeu :

  – nom
  – en-tête
  – description
  – fichiers associés (js, json, css) et leur liens
  – pied de page : licence

A utiliser en jeu :

  – nombre (ou liste) de canvas superposés
  – nombre maps et éventuellement niveaux (initialiser à 1)
  – tableau des persos suivant leur type avec surtout le nombre de persos à choisir
  – initialisation du score (et score max?)
  – nombre et types d'inventaires :
    – Sac de nourriture, liste des objets collectés, liste des ennemis tués, etc...
  – initialisation des objets interactifs à afficher dans les inventaires à 0

méthodes :

  – Afficher :
    – title (= nom du jeu)
      en haut (class header):
    – le nom du jeu
    – présentation
      à droite (class side)
    – la consigne pour le choix des persos
    – les images et les noms des persos à choisir
    – le numéro du niveau (correspondant à la map) initialisé à 1
    – le score (initialisé à 0)
    – l'inventaire avec les différents objets récupérés (div vide au départ)
      à gauche
    – le canvas
      en bas (class souscanvas)
    – la description du jeu
  – Taille du jeu
  – Règles
  – Touches utiles
    – les fichiers associés (js, json, css)
  – balises scripts
  – liens textuels vers les fichiers
    – le pied de page avec les licences et les auteurs (footer)
  – Remplir les différents tableaux :
– tableau des persos :
  – créer les persos un par un
  – les ajouter dans le tableau
  – Si le perso est interactif, donner sa classe
– Maps :
  – Créer les différentes maps de 1 à n...

objetPage (texte, bloc, image)

  – Type : texte, bloc, image
  – Nom
  – Taille
  – Id
  – Emplacement dans la page (id ou balise parente)
  – Class de l'emplacement (si changement ou transformation)
  – Pour un bloc :
    – balise
    – id
    – class (prévue dans la css ou transformant une classe existante)
    – contenu : html ou chaîne

- Pour une image :
  - Chemin
  - Nom de l'image (si différent du nom de l'objet)
  - Extension
  - Taille d'affichage (éventuellement calculée par une méthode)
- Pour du texte :
  - Fonte et taille du texte (à gérer par la transformation d'une class dans la css ?)
  - Chaîne

canvas :

- largeur
- hauteur
- emplacement x et y sur la page
- z-index du canvas

méthodes :

- afficher le canvas :
  - Prévoir les balises dans le HTML et une bordure dans la CSS
    - noter le z-index et en commentaire, à quoi chaque canvas est associé
  - Récupération du sélecteur dans le DOM
  - context

objetCanvas (texte, forme géométrique, image)

- Type : texte, forme, image
- Nom
- Emplacement de l'affichage (choix du canvas, coordonnées x et y)
- Pour du texte :
  - Fonte
  - Taille du texte
  - Chaîne
  - Couleur
- Pour une forme géométrique :
  - Type de forme
  - Taille
  - Coordonnées des points si besoin
  - Couleur remplissage
  - Couleur contour
- Pour une image :
  - Chemin
  - Extension
  - Taille
  - Échelle

imageTuile (image de type tuile ou perso)

Hérite d'objetCanvas, de type image

- Type : perso ou tuile (déterminera si on utilise la position du perso ou le numéro de la tuile)
- nom (du tileset ou de la spritesheet)
- Extension (du tileset ou de la spritesheet)
- chemin du dossier (du tileset ou de la spritesheet)
- largeur et hauteur (du tileset ou de la spritesheet)

Particularités :

- largeur de la tuile et hauteur de la tuile (extraite de la classe Tuile)
- position x et y sur le canvas (sera extrait du tableau de la map ou sera déterminé par la position du personnage)
- interactivité possible ? Si oui, nom de l'objet interactif (qui fera appel aux événements et fonctions choisis)
- Associé à une ImageDiv ? Si oui, donner son nom

méthode :

- Si l'imageDiv existe, instancier la classe correspondante pour pouvoir faire appel aux méthodes associées
- Charger le tileset et éventuellement l'imageDiv
- Choisir les instances en fonction du type :
  - Si le type est perso, instancier cette classe ;
  - Si le type est tuile, instancier la classe Map (qui gère le tableau des tuiles) et la classe Tileset (qui a une méthode pour numéroter les tuiles)
- calculer le nombre de lignes du tileset en fonction de la hauteur du tileset et de la taille de la tuile
- calculer le nombre de colonnes du tileset en fonction de la hauteur du tileset et de la taille de la tuile
- Pour les images qui sont des objets interactifs, appeler l'instance choisie

ImageDiv (à afficher hors canvas)

Cette image (ou vignette) ne peut faire partie d'un tileset ou d'une spritesheet même si ce qu'elle représente est une copie d'une des tuiles recadrée dans un éditeur d'image.

- Nom de l'image (ou suffixe à ajouter)
- Si elle représente la copie d'une tuile, numéro de la tuile associée (facultatif)
- Si elle représente la copie d'un perso, nom du perso associé (facultatif)

Particularités :

- Extension de l'image si différente de celle du tileset ou de la spritesheet
- chemin du dossier de l'image si différent de celle du tileset ou de la spritesheet
- Hauteur et largeur de l'image si différentes de celle du tileset ou de la spritesheet
- Identifiant du div où l'afficher (correspondant éventuellement à un inventaire)

méthodes spécifiques :

- Récupérer l'identifiant du div
- Afficher l'image dans le div
- Au besoin masquer ou supprimer l'image dans le div
- Si image cliquable (ou autre événement), lui ajouter le type d'événement (ex : onclick) et la fonction associée

map :

A l'heure actuelle map = niveau. On peut prévoir plusieurs niveaux (avec des nombres de perso différents par ex, des fonctions ou des événements supplémentaires...) ayant plusieurs maps. Le changement de niveau se fait alors en fonction du score, alors que le changement de carte se fait lors d'un événement de type : toucher les bords du canvas. Les caractéristiques de la map, en particulier les tableaux, sont dans un fichier json.

- numéro de carte (= numéro de niveau)
- couleur de fond (peut-être identique pour toutes les maps ou non)
- nom du tileset (peut-être identique pour toutes les maps ou non)
- instance du tileset utilisé (indiquera son nom)
- Nom des couches (dans un tableau?) : background, carte (couche interactive)
  - tableau(x) des numéros de tuile – Il peut y en avoir un par canvas lorsque ceux-ci sont superposés. Ils ont le même nombre de lignes et de colonnes que les tuiles sur le canvas.
  - Idée : Associer certains numéros de tuiles (si possible regroupés sur des lignes) ou un tileset précis (dont les numéros seraient décalés) à un canvas pour éviter de faire des tableaux pour chaque canvas ? ou donner des z-index (liés au canvas) différents à certaine tuiles ?
  - tableau avec numéros des tuiles et événements correspondant ou tableaux de numéros de tuiles pour chaque événement.
  - ex : tuile ou perso immobile pouvant être placé dans un inventaire

méthode :

- afficher la couleur de fond
- calculer :
  - nombre de lignes (= nombre de tuiles sur la hauteur du canvas)
  - nombre de colonnes (= nombre de tuiles sur la largeur du canvas)
- pour chaque couche : background, carte (couche interactive)
  - Appeler le fichier json
  - Parcourir le tableau pour dessiner au bon endroit chaque tuile (leur numéro a été attribué sur le tileset) : Le tableau est parcouru par lignes puis par colonnes. Pour chaque numéro, appeler la méthode de dessin de la tuile.
  - Au besoin, effacer la ou les différentes couches du canvas (par exemple avant un changement de map)

tileset :

voir imageTuile - Sa particularité est qu'on associe des numéros à chaque tuile présente sur le tileset, afin de reporter ces numéros dans le tableau de la map de différentes manières

- chemin du dossier
- nom du tileset
- Extension de l'image
- largeur et hauteur
- instance de tuile utilisée (indiquera sa taille)

méthode :

- Calculer nombre de tuiles sur une ligne du tileset : largeur / taille de tuile
- Numéroter les tuiles en fonction de la taille du tileset et de la taille des tuiles : A chaque largeur de tuile, on incrémente le numéro. Quand la largeur du tileset est atteinte, on descend sur l'axe y d'une hauteur de tuile et on reprend la numérotation.

tuiles :

voir imageTuile – dans ce cas, la largeur et la hauteur des tuiles sont identiques. Une tuile peut être gérée seule ou au sein d'une map (grâce à un tableau). Elle ne se déplace pas mais peut être animée ou non

- taille de tuile (largeur = hauteur)
- numéro (en fonction de l'emplacement sur le tileset) ou numéros s'il s'agit d'une tuile animée
- Type d'interactivité (plusieurs types sont possibles)

méthode :

- calculer la position x et y sur le tileset en fonction du numéro (ou des numéros pour une animation)
- dessiner la tuile sur le canvas : Son numéro définit ses positions x et y sur le tileset (voir méthode ci-dessus). Sa largeur détermine quelle portion du tileset découper. L'emplacement sur le canevas est déterminé par le tableau de la map.
- Au besoin effacer une tuile sur le canvas

perso :

voir imageTuile pour la partie affichage d'image dans le canvas et ImageDiv pour le choix du perso

- type (perso, pnj, adversaire, objet interactif non présent sur le tileset de la map)
  - Si perso joueur, une fonction permettra de l'afficher dans le div adapté et de le choisir parmi les différents persos pour affichage sur le canvas. Il sera associé aux événements clavier de déplacement.
  - Si objet interactif, sa particularité sera d'être immobile. Il est susceptible de se retrouver dans un inventaire s'il est attrapé (= objet interactif inventoriable)
  - si adversaire, il est susceptible de se retrouver dans un inventaire s'il est tué ou capturé (= objet interactif inventoriable)
- nom (identique à celui de la spritesheet) – Éventuellement ajout de catégories à prévoir (les différentes classes des RPG, les métiers...)
- largeur et hauteur
- position x et y (à initialiser)
- direction (en rapport avec la spritesheet et la touche enfoncée) : à initialiser
- Caractéristiques
  - Grâce à l'héritage, prévoir de pouvoir ajouter des caractéristiques précises, par ex numériques : points de vie, de force, de vitesse. Initialiser le tableau des caractéristiques à 0 pour un jeu simplifié.
  - Prévoir d'ajouter des éléments transportés par le perso : affichage (et animation) de l'objet relié à la position du perso en faisant un simple calcul lors de l'initialisation de la position de l'objet. ex : objet.pos = perso.pos +10
  - Type d'interactivité (plusieurs types sont possibles)

méthode :

– Choix du perso joueur :

– Afficher chaque perso disponible (image et nom) dans le div affiché sur le côté du jeu. Le nom de la vignette est différenciée de la spritesheet par un suffixe.

– repérer le clic sur le perso via onclick

– cela permet de faire le choix de la spritesheet à utiliser

– Afficher le perso choisi sur le canvas : afficher une partie précise de la spritesheet, correspondant à la largeur et la hauteur du perso. L'utilisation de la spritesheet convenable se fait en fonction du choix précédent ou via l'AI.

– déplacer le perso : changement position x ou y du perso – En fonction des touches de clavier (voir événements) si c'est un perso joueur ou de l'AI

– Placement du personnage (= Choisir la ligne sur la spritesheet) en fonction de la direction donnée par les touches du clavier ou de l'AI – voir méthode spritesheet

– animer le perso (= parcourir les colonnes de la spritesheet en boucle) – voir spritesheet. Quand il ne se déplace pas, le perso peut être fixe ou animé (il fait du surplace)

– Définir le nom des différents mouvements et y associer la plage de la spritesheet adaptée (colonnes et lignes)

– sélectionner un autre mouvement du perso (= choisir la plage de la spritesheet) en fonction des événements - voir méthode spritesheet

– Prévoir de pouvoir ajouter d'autres méthodes, en créant par exemple une liste vide de méthodes qu'on pourra ajouter ou non au jeu ?

– ex : changer la vitesse (caractéristique non encore implémentée) grâce à un événement

spritesheet

voir imageTuile – On travaille sur les numéros de lignes et de colonnes pour déterminer la direction dans laquelle regarde le perso (ex : une ligne par direction), et pour définir les différentes phases d'une animation.

– nom spritesheet identique à celui du perso

– Extension de l'image

– chemin du dossier de la spritesheet

– nombre lignes spritesheet : une ligne par direction

– nombre colonnes spritesheet : placer les différentes poses (et les différents mouvements?) sur la même ligne

méthode :

– pour la direction : définir le numéro de ligne de chaque direction

– pour les animations : définir la plage et le nombre de colonnes de chaque mouvement

– pour les différents mouvements : soit plage de colonnes et/ou de lignes différentes, soit spritesheet différente

Evénements

Type :

– Collision :

– Détection des bords du canvas en fonction de sa taille

– Provoque quelle fonction et sur quel objet ?

– ex : le personnage s'arrête

– ex : la map change (tileset et/ou tableau des tuiles)

– Détection de la position x et y du perso (initialisée dans la classe Perso et à retourner dans la méthode de déplacement)

– Déterminer un mode de calcul pour les rectangles de collision (sont-ils de la même taille que le perso?)

– Calcul du rectangle de collision autour de la position x et y en fonction de la taille du perso

Méthodes spécifiques :

– Détermination de la tuile sur laquelle le perso se trouve : la position d'une tuile est fonction de son numéro dans le tableau de la map

– Calcul du rectangle de collision de la tuile

– Comparaison des valeurs du perso et de la tuile

– Vérification de l'accessibilité de la tuile (peut-on marcher dessus?)

– Choix de la fonction à appliquer (suivant l'événement ou l'accessibilité)

– Si la tuile n'est pas interdite, son survol provoque-t-il une action ?

– Si la tuile est interdite, y a-t-il juste blocage du perso (qui ne peut plus avancer) ou une autre action est-elle prévu ?

– Mêmes opérations pour la collision avec un pnj

– clavier :

– écoute et détection des touches pressées ou relevées

– associer le numéro renvoyé à la valeur véritable de la touche pressée

– Choisir l'action à effectuer en fonction de cette touche

– clic (sur ou hors canvas) :

– détection de la position x et y au moment du clic et correction éventuelle si le clic est sur le canvas.

– Détection de l'id du div ou de l'élément DOM cliqué en dehors du canvas (grâce à onclick?)

– Choix de la fonction à lancer en fonction de l'emplacement cliqué

Fonctions associées:

– Affichage ou effaçage de tuiles, de persos ou d'images hors canvas

– Indiquer sur quel objet on agit en paramètre

– ex : Changement de carte : effaçage du canvas, changement de tableau de map et dessin sur le canvas

– ex : Choix perso : on change la spritesheet du perso affiché sur le canvas, l'effaçage du précédent n'est pas utile

– ex : Introduction d'un objet dans l'inventaire : on affiche une image dans le div d'inventaire – On l'efface du canvas et/ou on remplace par une autre tuile.

– Animation précise, direction, blocage d'un perso

– Indiquer en paramètre sur quel perso (ou objet mobile) on agit : joueur, pnj...

– ex : l'appui sur une touche directionnelle provoque un changement de direction lié au choix de la ligne sur la spritesheet

– ex : le passage (collision) sur une tuile « interdite » empêche les positions x et y de changer.

– ex : la collision avec un pnj provoque une animation ou un message textuel (à prévoir dans les classes et méthodes). L'animation ou le message peuvent impliquer un seul perso ou les deux.

– Changement et/ou affichage de valeurs numériques ou textuelles - L'affichage est prévu dans une autre fonction plus globale, sur le canvas ou sur la page.

– ex : Affichage du numéro (ou du nom) du joueur choisi.

La mise en valeur du numéro (ou du nom) peut-être effectuée par changement de la css, hors canvas.
– ex : changement du numéro du niveau, associée au changement de map

– ex : changement du nombre d'objet dans l'inventaire
– ex : changement du score, en fonction par exemple d'un nombre d'objets récoltés et placés dans l'inventaire

objetInteractif

– nom de l'objet
– Type de l'objet
– objetPage (ou imageDiv)
– Au besoin, id de l'inventaire où l'afficher
– objetCanvas (ou imageTuile)
– numéro(s) tuile ou position perso
– objet associé (si différent)
– numéro de tuile s'il correspondant à une tuile
– nom du perso s'il correspond à un perso
– nom de l'objet s'il s'agit un objetCanvas ou un objetPage
– caractéristiques de l'interactivité :
– inventoriable ? Si oui :
– id de l'inventaire
– lier l'introduction dans l'inventaire à un type d'événement (survol ou passage, clic, collision, touche clavier)
– nombre de points associé à son insertion dedans
– interdit – on ne peut pas marcher sur la tuile ou sur le perso
– contrôle collision
– animé ? Si oui, numéros des tuiles et prévoir l'appel à la fonction d'animation
– cliquable. Si oui :
– instanciation de l'événement clic
– id retourné (hors canvas)
– position x et y (sur canvas) :
– contrôler si la tuile ou le perso cliqué est bien à cette position
– appel à la fonction associée
– touche associée à l'objet (provoquant éventuellement un changement d'animation du perso)
– événement associé :
– ex : sera affiché dans le div de l'inventaire si on passe dessus (et/ou si on appuie sur une touche quand on passe dessus) : voir ImageTuile pour la partie affichage d'image dans le canvas et ImageDiv pour l'affichage dans l'inventaire
– ex : augmentation du score si on passe ou si on clique dessus (et/ou si on appuie sur une touche quand on passe dessus)
– ex : disparaît si on passe dessus (et/ou si on appuie sur une touche en même temps)
– ex : provoque le changement de map

Inventaire

Prévoir des copies des tuiles ou persos, recadrées grâce à un éditeur d'image, pour afficher en tant qu'imageDiv dans l'inventaire. Chaque inventaire est un div placé à un endroit sur la page, hors canvas.
– Nom de l'inventaire
– Id (si différent)
– Class (si besoin)
– id parent (facultatif)
– Affiché ou masqué
Type d'éléments pouvant être ajoutés
– Si les éléments sont des imageTuile : tableau contenant le numéro des tuiles ou le nom des persos pouvant y être ajoutés (ou tableau initialisé à 0 et ajout progressif des éléments en fonction de leur caractéristique, liée à un inventaire particulier)
– Si les éléments sont des images liées à une caractéristique ou à un score, lister ceux-ci
– Nombre d'éléments au départ et nombre d'éléments max
Méthodes
– insertion d'un élément dans l'inventaire (suite à un événement lié ou non à un objet interactif)
– suppression d'un élément de l'inventaire (suite à un événement lié ou non à un objet interactif)
– D'autres fonctions sont-elles appelées au moment où on place un objet dans l'inventaire ?
– ex : suppression ou remplacement d'une tuile ou d'un adversaire
– Les objets de l'inventaire peuvent-ils être sélectionnées (clic sur leur div ou sur un bouton, ou touche clavier associée) pour être ré-affichés sur le canvas ou supprimés ?

Points et scores

– Minimum et maximum (plage de chiffres autorisés)
– Initialisation (dépend du niveau, du perso)
– Types de points :
– Points accordés suite à un événement précis (clic, touche clavier, collision)
– Points accordés selon le nombre et le type d'éléments dans un inventaire
– Points accordés à un perso en fonction de ses caractéristiques (change selon le perso choisi)
– affichage d'une ou plusieurs imageDiv dans un inventaire prévu à cet effet (par exemple un nombre d'étoiles ou de cœurs)
– Score calculé à partir de tous ceux-là
– Nombre de points associés à l'insertion d'un élément dans l'inventaire, en fonction de son type ou de ses caractéristiques
– Atteinte d'un nombre de points précis et action associée (dépend du niveau, du perso)
– ex : indiquer que l'inventaire est plein (message textuel ou changement css...)
– ex : changement de niveau (= changement de map, pour l'instant)
– ex : affichage d'une imageDiv dans un inventaire prévu à cet effet
– ex : affichage d'un écran gagnant sur le canvas (changement de map ? Nettoyage des canvas et remplacement de la couleur de fond par une image pleine ? Rectangle et texte ?)
Méthodes :
– Calculer le score initial ou à atteindre en fonction de caractéristiques précises (dépend du niveau, du perso)
– Augmenter ou diminuer le score en fonction des événements. Retourner ce chiffre
– Associer une fonction lorsqu'un nombre de points précis est atteint
– Afficher ou masquer le score ou certains types de points (texte dans un div ou texte sur le canvas, en utilisant la méthode appropriée)