Mario

W2 – JS Avancé

Vous aimez les aventures de Mario ? Et la POO ? Nous on les adore autant l'une que l'autre !!

Description du document

Titre	Mario
Date	17/08/2017
Auteur	Odin Duclos
Responsable	Sophie Viger
E-mail	odin1.duclos@epitech.eu
Sujet	Jeu vidéo Mario
Mots-clés	Jeu vidéo, Mario, JS, POO
Version	1.0

Tableau des révisions

Date	Auteur	Sections	Commentaire
17/08/2017	Odin Duclos	Toutes	Création du document.

Modalités de rendu

Serveur	Git epitech
Authentification	Blih
Dépôt	mario
Droits pour Lyon	ramassage-tek :r odin1.duclos@epitech.eu:r
Droits pour Paris	ramassage-tek :r thomas1.fortassin@epitech.eu:r

Sommaire

Introduction

Bowser a encore capturé la princesse Peach, c'est un comble ! Mais cette fois les choses sont différentes. En effet, la princesse porte sur elle les micro films de vos bibliothèques JS. Votre mission, si vous l'acceptez, est donc de récupérer ces microfilms en permettant à Mario de se déplacer dans un monde que vous aurez créé en JS pur afin de récupérer vos bibliothèques des griffes de l'immonde Bowser, et accessoirement de récupérer Peach (encore). La vengeance est un plat qui se mange froid !

Notions

- JS
- POO

Restrictions

Les restrictions suivantes sont appliquées à votre projet, toute contravention entrainant un 0 et la non correction de votre projet :

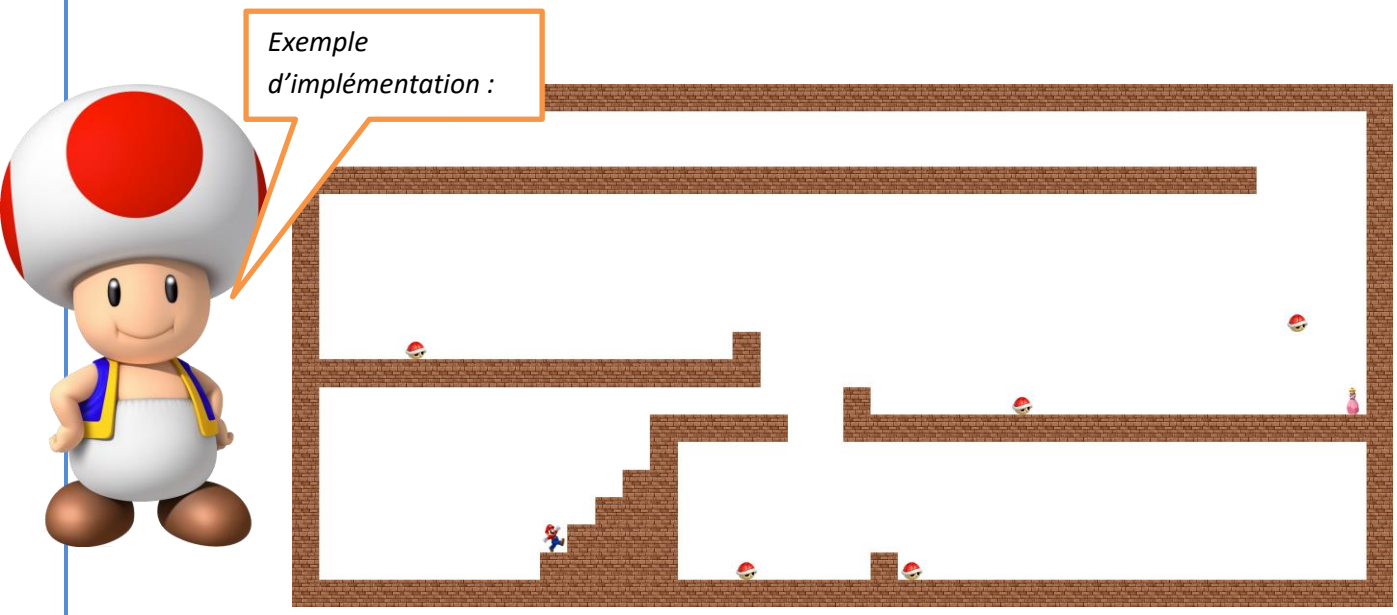
- Aucune bibliothèque, plugin, ou module n'est toléré au cours de ce projet
- Votre projet doit être intégralement développé en Javascript orienté objet
- Vous devez respecter les classes fournies dans les spécifications

Checkpoint : Attention, avant de commencer le projet vous devez vous renseigner sur la POO en JS. Et avoir lu intégralement le cours Mozilla sur le sujet[1].

[1] https://developer.mozilla.org/fr/docs/Web/JavaScript/Introduction_%C3%A0_JavaScript_orient%C3%A9_objet

Procédure

Dans ce sujet, vous devez créer un jeu vidéo simple où Mario peut se déplacer de cases en cases, sauter sur des koopas pour les tuer, ou mourir heurté par des koopas. Le jeu prend fin quad Mario rattrape Peach ou qu'il a assouvi sa vengeance en tuant tous les koopas.



Pour cela votre chef de projet toad vous a fourni un cahier de spécifications que vous devrez respecter à la lettre lors de la présentation de votre projet à l'équipe technique des toads.

Les fonctionnalités

Au minimum votre projet devra permettre :

- De lancer une partie avec la map passée en paramètre
- De se déplacer avec les flèches du clavier de 1 case vers la gauche ou vers la droite en 1 animation minimum pour une durée totale de 100ms
- De sauter avec la barre d'espace de trois cases vers le haut en 3 animations minimum pour un total de 300ms
- Les koopas devront se déplacer de gauche à droite jusqu'à rencontrer un mur puis changer de sens
- Quand mario est heurté ou heurte un koopa, il doit disparaitre et un écran de défaite être affiché
- Quand mario saute sur un koopa, le koopa doit disparaitre
- Quand mario a éliminé tous les koopas, ou qu'il a atteint peach un écran de victoire doit apparaitre
- Tous les objets autres que les murs doivent tomber de 1 case vers le bas en 1 animation minimum pour un total de 100ms jusqu'à rencontrer un mur/sol

Spécifications

Vous devez au minimum respecter les spécifications suivantes :

```
//
https://developer.mozilla.org/fr/docs/Web/JavaScript/Introduction_%C3%A0_JavaScript_orient%C3%A9_objet
var Cell = function (y, x, image) {
    this.x = x;
    this.y =  y;
    this.image = image;
    // crée un élément img et l'insère dans le DOM aux coordonnées x et y
    this.update = function () {
        // met à jour la position de la cellule dans le DOM
    };
    this.checkCollision = function (cell) {
        // retourne true si la cellule est aux même coordonnées que cell
    };
    this.die = function () {
        // cell est retiré de la map
    };
};

var Mario = function (y, x, image) {
    // Mario hérite de Cell
    this.falling = false;
    this.input = new Input(['ArrowLeft', 'ArrowRight', 'Space']);
    this.jump = {
        power: 0, // hauteur du saut en nombre de cellules
        interval: null // identifiant de l'intervalle de temps entre chaque animations du saut
    };
    this.makeJump = function () {
        // mario monte d'une case s'il le peut et s'il lui reste du power
        // s'il ne le peut pas, il met fin à l'intervalle de temps entre chaque animation du saut
        // mario met à jour le dom à chaque animation de saut
        // si mario saute dans un koopa, il meurt
    };
    this.fall = function () {
        // mario se déplace d'une cellule vers le bas s'il le peut et met falling à true
        // si mario tombe sur un koopa, il meurt
    };
    this.die = function () {
        // mario met fin à son intervalle d'animations
        // mario est retiré de la map
    };
    this.move = function () {
        // si l'Input est flèche de gauche, mario se déplace à gauche s'il le peut
        // si l'Input est flèche de droite, mario se déplace à droite s'il le peut
        // si l'Input est espace, mario commence un saut
        // si mario rencontre un koopa après son déplacement, il meurt
    };
    this.interval = setInterval(function () {
        mario.fall();
        mario.move();
        mario.update();
    }, 100);
```

```
};

var Koopa = function (y, x, image) {
    // Koopa hérite de Cell
    this.direction = 'left';
    this.die = function() {
        // koopa met fin à son intervalle d'animations
        // koopa est retiré de la map
    };
    this.move = function () {
        // koopa se déplace en direction de direction s'il le peut
        // sinon il change de direction
        // si koopa recontre mario, mario meurt
    };
    this.fall = function () {
        // koopa se déplace d'une cellule vers le bas s'il le peut
    };
    this.interval = setInterval(function () {
        koopa.fall();
        koopa.move();
        koopa.update();
    }, 200);
}

var Input = function (keys) {
    this.keys = {};
    // Input récupère les touches actives du clavier
}

var Map = function (model) {
    this.map = [];
    this.generateMap = function () {
        // instancie les classes correspondants au schema
        // avec :
        //      w => Cell
        //      k => Koopa
        //      m => Mario
    };
    this.checkCollision = function (cell) {
        // parcourt la map et renvoie la cellule aux mêmes coordonnées que cell
    };
    this.delete = function (cell) {
        // retire la cell de map
        // retire la cell du dom
        // delete la cell
    };
};

var schema = [
    'wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww',
    'w                                      w',
    'w                                 k    w',
    'wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww    w',
    'w                                      w',
    'w                                      w',
    'w                                      w',
    'w                                      w',
    'w                                      w',
    'w          k    w                      w',
```

```
    'wwwwwwwwwwwwwwwww                      w',
    'w                   w           k      w',
    'w            wwwww  wwwwwwwwwwwwwwwwwwww',
    'w            w                         w',
    'w           ww                         w',
    'w          www                         w',
    'w         wwww                         w',
    'wm       wwwww k     w      k         pw',
    'wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww'
];
var map = new Map(schema);
map.generateMap();
```

Attention ! Toute modifications de la structure du code devra être justifiée en soutenance et pourra être considérée comme un non-respect des consignes.

Information : N'oubliez pas que dans les fonctions de callback le contexte (this) peut changer et se rapporter à l'élément sur lequel est bindé la fonction (typiquement windows sur les listeners).

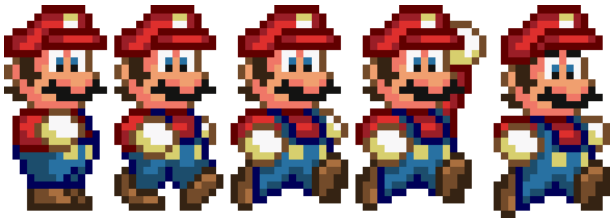
Et la suite ?

Vous avez à présent le squelette de votre jeu, mais il est temps d'aller plus loin ! Pour la suite du projet, choisissez et implémentez une des trois fonctionnalités ci-dessous. Si vous choisissez d'en implémenter plus d'une, celles-ci seront considérées comme des bonus.

Les animations

Dans cette partie, vous devez implémenter un système d'animations poussées pour vos objets mobiles :

- Les koopas et Mario doivent se déplacer de cases en cases en plusieurs animations.
- Vous devez implémenter un système de sprite pour Mario afin de différencier la position de saut, la position de repos, et les deux positions de course (pied gauche en avant ou pied droit en avant) au minimum pour les sens de course droite et gauche. Exemple :



- Vous devez implémenter un système de sprite pour les koopas afin de simuler un glissement avec rotation horizontale (au moins deux animations de rotation de 90), et la mort du koopa. Exemple d'animation de rotation horizontale :

Les bonus en jeu

Vous devez implémenter plusieurs systèmes de bonus modifiant le comportement du jeu.

L'étoile

Mario doit changer de couleur, se déplacer plus vite, et être invincible pendant 10 secondes après avoir heurté une étoile.

Le champignon

Mario doit gagner une vie supplémentaire en heurtant un champignon.

Le ballon

Mario se transforme en Mario Ballon en heurtant un ballon, il peut alors profiter d'une poussée verticale infinie pendant les sauts. Tant que le joueur reste appuyé sur espace, Mario continue à monter.

L'éditeur de carte

Vous devez implémenter un éditeur de carte user-friendly. Pour cela, le joueur doit pouvoir créer une nouvelle carte, la sauver, la modifier, ou la supprimer.

Lors de l'édition, il peut glisser déposer des blocs sur la carte parmi une liste de blocs proposés.

Le boss

Vous devez implémenter Bowser. Ce boss fait deux cases sur deux et peut se déplacer à gauche ou à droite, ou sauter, de façon à tenter de tuer Mario en lui rentrant dedans. Pour tuer Bowser, Mario doit lui sauter trois fois dessus.