Projet Web Frontend Hetic

contact@ls-a.fr
https://www.ls-a.fr

Table des matières

Contexte

Vous avez été missionné pour réaliser un prototype de jeu de type "Runner" avec les technologies du web : HTML, CSS et JavaScript.

Les jeux de type "Runner" mettent en scène un personnage avançant de plus en plus vite dans un niveau possédant des plate-formes, personnage que le joueur ne peut pas stopper : il ne reste plus qu'à éviter les obstacles.

Plus d'informations :
https://fr.wikipedia.org/wiki/Jeu_de_plates-formes#Jeu_de_course_sans_fin

Fonctionnalités demandées

Votre jeu de runner devra disposer des fonctionnalités suivantes :

- Pour la partie "Runner" :
  - Charger un niveau à partir d'un fichier
  - Afficher un personnage
  - Faire avancer ce personnage à une certaine vitesse dans un niveau en deux dimensions
  - Faire sauter ou s'accroupir ce personnage
  - Décompter des points au fur et à mesure de l'avancement du personnage dans le niveau
  - Afficher le score final en cas de victoire ou d'échec
  - Changer la vitesse d'avancement du personnage dans le niveau
- Pour la partie "Edition" :
  - Créer un nouveau niveau
  - Modifier un niveau existant
  - Supprimer un niveau
  - Placer des blocs au sein du niveau
  - Sauvegarder le niveau dans un fichier

Le format de fichier

Un niveau est stocké dans un fichier au format JSON. Ce format contient une liste de blocs de différents types, les uns à la suite des autres.
On retrouvera différents "blocs" :

- Une ligne droite (pas d'obstacle)
- Un obstacle au sol
- Un obstacle en hauteur

Chaque bloc mis bout à bout constitue le niveau. Une fois le fichier lu, l'interface du niveau apparaît, avec les lignes droites et les obstacles. C'est avec les informations de ce fichier que vous pourrez faire avancer le personnage dans le niveau, gérer les collisions pendant une partie et ainsi déterminer les conditions de fin de partie (victoire, défaire, score).

Pour mieux identifier les niveaux du jeu, il est recommandé d'enregistrer les niveaux dans un fichier ".jmpr". Exemple de nom de fichier de niveau : "Hello-World.jmpr".

Voici la syntaxe **à respecter** pour le fichier .jmpr :

```
{
  "title": "Nom du niveau",
  "creator": "John Doe",
  "difficulty": 1,
  "blocks": [
    {"type": "A"},
    {"type": "B"},
    ...
  ]
}
```

Les champs sont les suivants :

- title : Le titre du niveau (affiché dans l'interface)
- creator : Le nom de la personne ayant créé le niveau (affiché dans l'interface)
- difficulty : La difficulté ressentie par les joueurs, nombre de 1 à 5 (idem)
- blocs : La liste des blocs, contenant pour chacun le "type" du bloc.

Type des blocs :

- A : Ligne droite
- B : Obstacle au sol
- C : Obstacle en hauteur

L'affichage

Dans un jeu de type "Runner", le personnage ne se déplace pas vraiment : c'est tout le niveau qui se déplace sous le personnage. Une fois le fichier de niveau chargé, placez le personnage dans un bord de l'écran, placez les blocs les uns à la suite dans l'interface, et faites dérouler l'ensemble des blocs sous le personnage à une vitesse vous semblant cohérente. Cette vitesse dépendra directement de la taille que représente chaque bloc à l'écran.

Concernant la représentation visuelle des blocs, vous pouvez les générer en HTML/JavaScript. Vous pouvez utiliser des images libres de droit sur internet, créer vos propres images ou même utiliser des blocs de couleurs unies. L'esthétique générale du jeu découlera de vos choix, mais le point le plus important reste que le jeu fonctionne.

Difficulté, vitesse d'avancement et score

Le jeu propose différement niveaux de difficultés :

- Découverte
- Normal
- Perfectionnement
- Expertise
- Maîtrise

Ces niveaux de difficulté correspondent aux classiques "Facile" "Moyen" "Difficile" et ainsi de suite. Changer le niveau de difficulté du jeu, c'est changer la vitesse de défilement dans les niveaux. Associez à chaque niveau de difficulté un multiplicateur de la vitesse de défilement, qui se retrouvera pendant le jeu.

De plus, tous les **"N"** blocs, la vitesse augmentera graduellement dans le niveau. Ce nombre "N" est directement dépendant de la taille de vos blocs sur l'écran, choisissez cette valeur avec attention.

Le score est calculé par rapport aux nombres de blocs passés, au multiplicateur de vitesse actuelle et à des points accordés à chaque bloc. Pour ce projet, le nombre de points par bloc est imposé à **10 (dix)**.

Le score est calculé de la manière suivante :

```
score = score + (multiplicateur_vitesse * points_bloc)
```

En cas de victoire, le score est augmenté de cette manière :

```
score = score + (nombre_blocs_du_niveau * multiplicateur_vitesse_finale)
```

Premier exemple :

Démonstration

Niveau réussi, contenant 13 blocs. Vitesse de base à x1.2 augmentant de 0.1 tous les 10 blocs

- Score 10 premiers blocs : (10 x 1.2 x 10) = 120
- Score 3 derniers blocs : (3 x 1.3 x 10) = 39
- Score victoire : 13 x 1.3 = 16.9

Score définitif : 120 + 39 + 16.9 = 175.8 (arrondi à 176)

Deuxième exemple :

Démonstration

Niveau réussi, contenant 265 blocs. Vitesse de base à x1.6 augmentant de 0.2 tous les 20 blocs

- Score blocs 1 à 20 : (20 x 1.6 x 10) = 320
- Score blocs 21 à 40 : (20 x 1.8 x 10) = 360
- Score blocs 41 à 60 : (20 x 2.0 x 10) = 400
- ...
- Score blocs 241 à 260 : (20 x 4 x 10) = 800
- Score blocs 261 à 265 : (5 x 4.2 x 10) = 210
- Score victoire : 265 x 4.2 = 1113

Score définitif : somme de tous les calculs ci-dessus

Troisième exemple :

Démonstration

Niveau échoué, contenant 60 blocs mais seulement 23 de passés. Vitesse de base à x1.4 augmentant de 0.25 tous les 10 blocs

- Score blocs 1 à 10 : (10 x 1.4 x 10) = 140
- Score blocs 11 à 20 : (10 x 1.65 x 10) = 165
- Score 21 à 23 : (3 x 1.90 x 10) = 57

Score définitif : 140 + 165 + 57 = 362

Contrôles

La partie "Runner" contiendra peu de possibilités concernant les contrôles :

- Eviter un obstacle en bas (sauter)
- Eviter un obstacle en haut (s'accroupir)
- Mettre en pause

Pendant la pause, voici les choix disponibles :

- Reprendre (bouton ayant le focus par défaut)
- Recommencer
- Quitter le niveau

Pendant la fin de partie, voici les choix disponibles :

- Recommencer
- Quitter le niveau (bouton ayant le focus par défaut)

> **Notes**
>
> Dans les menus "Pause" et "Fin de partie", un bouton aura le focus par défaut. C'est le bouton qui, si vous appuyez sur la touche "Entrée" ou "Espace" d'un clavier, sera actionné. A chaque appui sur la touche "Tab" de votre clavier, le focus se déplacera sur le bouton suivant. A chaque raccourci "Shift+Tab" de votre clavier, le focus se déplacera sur le bouton précédent.

En plus de l'interface associée à ces différentes actions (popup, flou ou autres effets visuels), il faudra associer à chaque action un événement (touche ou souris). Cette association est laissée à votre réflexion. Vous pouvez également associer d'autres touches du clavier à différentes actions, comme déplacer le focus sur le bouton suivant ou précédent.

Qu'importe la solution que vous aurez choisi, il faudra permettre dans un menu "Paramètres" le changement des touches associées à chacune des actions : votre jeu doit permettre aux utilisateurs ayant des difficultés avec vos contrôles de les associer à quelque chose de plus simple.

Exemple 1 : Clavier

Démonstration

- Débuter la partie : touche du clavier "G"
- Eviter un obstacle en bas (sauter) : touche du clavier "I"
- Eviter un obstacle en bas (s'accroupir) : touche du clavier "K"
- Mettre en pause : touche du clavier "P"
- Reprendre (dans le menu pause) : touche du clavier "P"
- Recommencer (menu pause ou fin de partie) : touche du clavier "R"
- Quitter le niveau (menu pause ou fin de partie) : touche du clavier "Q"

Exemple 2 : Clavier + Souris

Démonstration

- Débuter la partie : clic gauche
- Eviter un obstacle en bas (sauter) : clic gauche
- Eviter un obstacle en bas (s'accroupir) :clic droit
- Mettre en pause : touche du clavier "Echap"
- Reprendre (dans le menu pause) : touche du clavier "Echap"
- Recommencer (menu pause ou fin de partie) : touche du clavier "R"
- Quitter le niveau (menu pause ou fin de partie) : touche du clavier "Q"
- Changer le focus vers l'élément suivant (dans un menu) : Molette vers le bas
- Changer le focus vers l'élément précédent (dans un menu) : Molette vers le haut

Editer un niveau

Un autre mode de jeu consiste à pouvoir éditer un niveau. Cet éditeur doit permettre d'ajouter les différents blocs (ligne droit, obstacle bas, obstacle haut) à la suite les uns des autres. Il serait souhaitable d'ajouter différentes fonctionnalités :

- Prévisualiser le niveau en cours de construction
- Supprimer un bloc (et ainsi décaler tous les suivants)
- Afficher le nombre de blocs total du niveau
- Afficher le score maximum du niveau dans chacune des difficultés
- Importer un niveau déjà existant

Une fois le niveau réalisé, il faudra permettre à l'utilisateur d'exporter son niveau dans un fichier ".jmpr" contenant le JSON du niveau.

L'éditeur permettra également de fournir les autres informations du niveau : nom, difficulté et auteur du niveau.

Bonus

Voici quelques idées supplémentaires à implémenter dans le projet. Si certaines sont réalisées avec succès, un bonus pourra être appliqué sur votre note :

Ajouter des blocs contenant des bonus :
Créez des bonus influençant les mécaniques de jeu. Voici ceux à ajouter :

- Bonus "Terraformation" : remplace les **N** blocs suivants par des blocs "ligne droite"
- Bonus "Vitesse" : augmente la vitesse et donc le score, tout en rendant le jeu plus difficile
- Bonus "Multiplicateur" : pour les **N** blocs suivants, le score de ces blocs est doublé
- Bonus "Bouclier" : en cas de collision avec un bloc, "perd" le bouclier mais la partie continue
- Bonus "Score" : augmente le score de **N** points de manière fixe

Ces bonus sont positionnés sur le niveau de manière déterminée, à l'avance (aucun aléatoire). Ils s'obtiennent sur le même principe de collision que les obstacles. Si un bonus est en bas, ne rien faire permet d'obtenir le bonus. Si un bonus est en haut, il faudra sauter au bon moment pour obtenir le bonus.

Ajoutez ces blocs ("ligne droite+bonus X", "obstacle bas+bonus Y", ...) dans l'éditeur pour pouvoir les positionner sur le niveau. Dans le fichier ".jmpr", le type de ces blocs sont les suivants :

- A : Terraformation
- B : Vitesse
- C : Multiplicateur
- D : Bouclier
- E : Score

Il faudra également ajouter la position du bonus (haut ou bas), représenté dans le type par "1" (bas) ou "2" (haut). Ainsi, un bloc "ligne droite" avec un bonus "Vitesse" en bas aura pour type "AB1" (A pour ligne droite, B pour vitesse, un pour "en bas").

Ajouter une ambiance aux niveaux :
Ajoutez dans le JSON du fichier ".jmpr" une clef "assets" comme suit :

```
{
  "title": "Nom du niveau",
  "creator": "John Doe",
  ...
  "assets": {
      "background": "https://...",
      "A" : "https://...",
      "B" : "https://...",
      ...
  }
}
```

Chacune des clefs de "assets" correspondra à l'image associée à chacun des blocs. Ainsi, un bloc "A" par défaut aura la représentation par défaut du jeu (exemple : un carré rouge), mais si "assets" est défini et si la clef "A" est définie dans "assets", alors à la place de la représentation par défaut, l'image déclarée doit être utilisée.
Ajoutez également une clef "background" dans "assets" pour faire un fond défilant pour votre jeu.

Ajoutez de la musique :
Toujours dans la clef "assets" du JSON du fichier ".jmpr", ajoutez une clef "melody" comme suit :

```
{
  "title": "Nom du niveau",
  "creator": "John Doe",
  ...
  "assets": {
      "melody": "https://...",
      ...
  }
}
```

La musique référencée devra être jouée en boucle pendant le niveau.
Une fois ceci réalisé, ajoutez des effets sonores à la fin de la partie, avec deux mélodies : une pour la victoire et une pour la défaite. Ces deux effets sonores ne doivent pas faire partie du fichier ".jmpr".

Modalités

Travail en groupe

Ce projet devra être réalisé en groupe de 6 à 8 personnes. La composition des groupes est à votre convenance.

Les groupes sont à renseigner dans le Google Doc prévu à cet effet : https://docs.google.com/spreadsheets/d/1nkWG8TI912BaufcMhzbwcaTI0rRaQfHpsJ62RRurxQM/edit?usp=sharing.

Contraintes techniques

- Votre rendu devra être effectué dans une archive ZIP envoyée par mail, nous vous conseillons donc une taille inférieure à 25Mo (voir livrables).
- Les seuls langages autorisés pour ce projet sont HTML, CSS et JavaScript.
- Vous n'êtes pas autorisés à utiliser NodeJS.
- Les frameworks et les bibliothèques ne sont pas autorisés.

Le rendu du projet

Le projet doit être rendu avant le 19 février 2023 à 23h59, à l'adresse damien@ls-a.fr. Votre rendu devra être accompagné de la liste des membres de votre groupe. Ce rendu devra être réalisé sous la forme d'un ZIP en pièce jointe du mail. Attention donc à la taille de votre rendu. Les rendus par liens (WeTransfer, Git, OneDrive, etc.) ne seront pas pris en compte.

Contraintes de nommage

- L'archive ZIP devra porter le nom de votre groupe ; par exemple `Groupe08.zip` pour les membres du groupe 8.
- Le mail devra être envoyé avec l'objet "Projet Frontend - GroupeXX", où XX représente votre numéro de groupe ; par exemple `Projet Frontend - Groupe08` pour les membres du groupe 8.

Bonus potentiels

L'ajout de fonctionnalités supplémentaires ou un travail particulièrement qualitatif sur un élément du projet peut donner droit à des points supplémentaires. Les bonus détaillés dans la section associée, s'ils sont réalisés avec succès, donneront droit à des points supplémentaires.

Malus potentiels

Des malus pourront être appliqués, notamment :

- En cas de copier/coller trop important d'une source, que ce soit internet ou un autre groupe
- Retard d'envoi du projet
- Hors sujet
- ...