Epic Wheel Hero

Documentation technique

DIVTEC | Gallucci Alex & Steven Halaghia

Table des matières

[Introduction 3](#_Toc49842110)

[Les notes 3](#_Toc49842111)

[Disposition 3](#_Toc49842112)

[Créations des scripts 3](#_Toc49842113)

[Mis en fonctions des notes 3](#_Toc49842114)

[Les roues 5](#_Toc49842115)

[Le viseur 5](#_Toc49842116)

[Thèmes 6](#_Toc49842117)

[Autres fonctionnalités 6](#_Toc49842118)

[Les vies 6](#_Toc49842119)

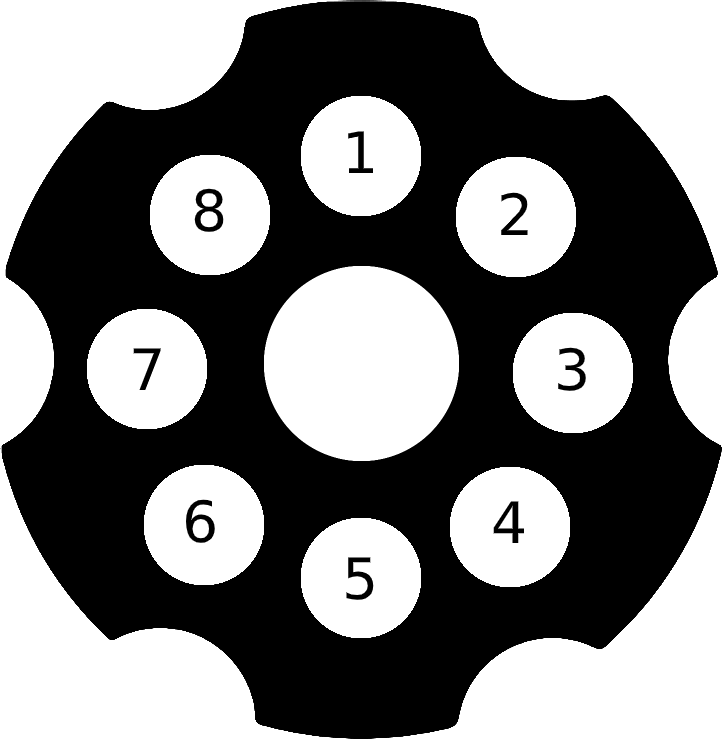
[Le score 6](#_Toc49842120)

# Introduction

Avant de procéder à la lecture de cette documentation technique, assurez-vous d’avoir lu le « Guide d’utilisation » de ce même jeu.

# Les notes

## Disposition

Les notes se déplacent de case en case dans le sens des aiguilles d’une montre. Ils apparaissent à la case 5, se déplacent de cases en case jusqu’à revenir à la même case 5 ou ils seront détruits.

## Créations des scripts

Pour gérer les notes de chaque roue, 2 fichiers scripts seront crées : mouvementRoueG.cs et mouvementRoueD.cs.

Une fois les 2 scripts crées,

Le fichier mouvementRoueG.cs sera lié à la note de la roue Gauche.

[Insérer image ici]

Le fichier mouvementRoueD.cs sera lié à la note de la roue Droite.

[Insérer image ici]

## Mis en fonctions des notes

Les notes sont tout d’abord **déclarées** dans le fichier mouvementRoueG.cs et mouvementRoueD.cs.



L’interface Unity affiche désormais 6 nouvelles options. Il faut renseigner quelles images correspond à quelle note.

[insérer image ici]

Pour que les notes puissent se déplacer, elles ont besoins de positions cibles. Pour cela nous allons créer dans chaque cellules de la roue, un objet qui sera rendu invisible. On implémente le code suivant dans nos 2 fichiers mouvementRoue.



Nous pouvons désormais initialiser ces cibles. Nous faisons cela sous la fonction Awake() de Unity.



Nous allons par la suite beaucoup travailler avec la propriété transform des objets justes crées.

Afin de nous simplifier la tâche, nous déclarons directement 8 nouvelles variables qui seront les notes initialisées avec l’attribut transform.



On donne à ces objets une dimension.



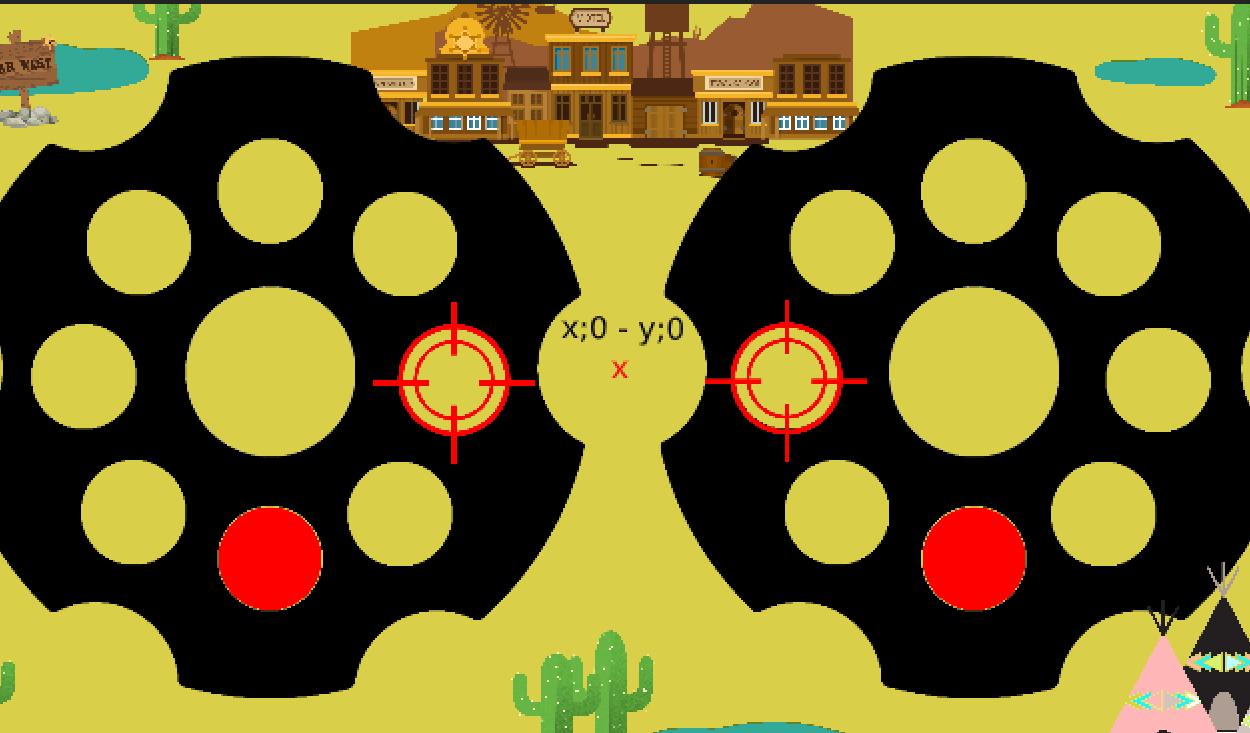
Place les 8 objets dans leur emplacement respectif dans la roue.



# Les roues

Les roues représentent des chargeurs de révolver. Elles sont parfaitement fixes, seuls les notes à l’intérieur se déplacent de cases en cases.

Celles-ci sont positionnées de sorte à ce que les points à l’exacte milieu de celles-ci soient x=0, y=0.



Les points entre les roues sont X;0 , Y;0

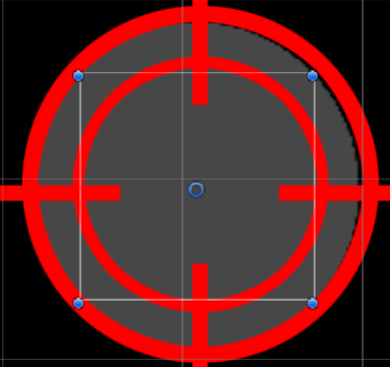
La roue de gauche se positionne aux coordonnées X;6.2 , Y;0.

La roue de droite se positionne aux coordonnées X;-6.2 , Y;0

# Le viseur

Afin de savoir si la note est parfaitement dans le viseur, nous utilisons le système suivant :

4 images invisibles sont disposées à l’intérieur du viseur.



Quand le joueur appui sur une note les évènements suivant se passent :

|  |  |  |
| --- | --- | --- |
| Illustration | Condition(s) | Conséquence(s) |
|  | Les 4 points touchent la note. |  |
|  | Moins que 4 points touchent la notes (0-3) |  |

# Matériel / logiciel

## Matériels nécessaire

|  |  |  |
| --- | --- | --- |
| 2x | Guitare Hero pour Xbox One | Une image contenant guitare, assis, table  Description générée automatiquement |
| 1x | Moniteur / Ecran pour l’affichage | Une image contenant moniteur, équipement électronique, télévision, plat  Description générée automatiquement |
| 2x | Adaptateur USB pour Guitare Hero | Une image contenant téléphone mobile  Description générée automatiquement |
| 1x | Enceinte / Haut-parleur | Une image contenant équipement électronique, moniteur, assis, ordinateur  Description générée automatiquement |
| 1X | PC / laptop  Windows | Une image contenant équipement électronique, moniteur, assis, ordinateur  Description générée automatiquement |

## Logiciels nécessaires

|  |  |  |
| --- | --- | --- |
| 1X | Epic Wheel Hero | [insérer le lien ici] |
| 1X | JoyToKey | https://joytokey.net/en/download |

## Fichiers nécessaires

|  |  |  |
| --- | --- | --- |
| 1x | EpicWheelHero.cfg |  |

# Thèmes

Far West

Le premier thème du jeu, celui qui s’affiche quand on lance la partie, est un thème Far West.

Celui-ci est composé de :

* 1x image d’arrière plan
* 4x cactus
* 1x panneau en bois
* 1x ville far west
* 1x trio de tepee
* 1x lac

Star Wars

# Autres fonctionnalités

## Les vies

Quand la partie commence, chaque joueur possède 3 vies.

Le joueur perd une vie si :

* Il appuie sur la note avant qu’elle n’arrive dans le viseur.
* Il appuie sur la note après qu’elle soit arrivée dans le viseur.
* Il appuie sur une note spéciale « perte d’une vie ».

## Le score

### Explications

Un score est constamment affiché à l’écran pour chaque joueur. Pour chaque note « classiques » touchées, le score augmente de 10 points.

Quand le joueur touche plusieurs touchées d’affiée, il déclenche **un combo**.

Celui-ci fonctionne de la manière suivante :

|  |  |
| --- | --- |
| Nombre de notes touchée d’affilée | Score gagné |
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |
| 5 et + | 50 |

Une note spéciale « +100 » fera augmenter le score de 100 points si touchée.

Il n’y a pas de limite maximale au score.

### Implémentation

Un scripte nommé « scoreScript G.cs » est assigné à la roue de gauche.

Un scripte nommé « scoreScript D.cs » est assigné à la roue de droite.

Les 2 classes sont composées du même code :



La variable scoreValue contient le score sous forme d’entier.

L’objet score est l’editText, ici de type TextMeshProGUI.

Il ne reste plus qu’à assigner l’editText au bon composant (dans la fonction start), et à recharger à chaque image par seconde le score (dans la fonction Update).

Désormais, a chaque fois que l’on manipulera le score, on utilisera la variable score de ce script -> scoreScriptG.scoreValue = x;.

Il en va de même pour la roue de droite avec scoreScriptD.scoreValue = x;.