Guitar Hero

version 2

DIVERSITY by EPITECH

I. Introduction

Le Lua a été créé en 1993 au Brésil afin de pouvoir être embarqué au sein d'autres applications par trois étudiants d'une université de Rio de Janeiro.

Il est utilisé pour faire de l'embarqué mais surtout du réseau et du jeu vidéo. Il a notamment été utilisé dans Roblox, World of Warcraft, Garrysmod et bien d'autres. Il est également utilisé dans des moteurs de jeu vidéo tels que CryENGINE.

Pour ce qui est de Love2D c'est un moteur de développement de jeu vidéo. On le retrouve fréquemment dans des compétitions de développement de jeux vidéo.

Il a notamment été utilisé pour créer le célèbre jeu « Move or Die » ainsi que « Blue Revolver » ou encore « Oh my giraffe ».

LÖVE®

II. Consignes

- Le nom du repo est : guitarHero_cc
- En cas de questions, n'hésitez pas à vous entraider et si vous êtes toujours bloqué, un cobra se fera une joie de vous aider.
- Vous allez utiliser le langage de programmation « Lua » sur la plateforme Love2D
- Si l'installation de Love2D ne se déroule pas bien, recommencez et demandez de l'aide à un cobra

III. J'aime la guitare mais la guitare ne m'aime pas

Vous êtes au début des années 2000. Influencé par la culture punk rock et des ambiances de skate parc comme dans les jeux vidéo de la franchise Tony Hawk, vous décidez de vous mettre à la guitare.

Problème, vous aimez la guitare mais vous n'êtes franchement pas doué. Vous vous dites donc que vous allez créer un jeu vidéo afin de pouvoir vous entrainer plus rapidement et ainsi enfin maitriser la guitare à la perfection.

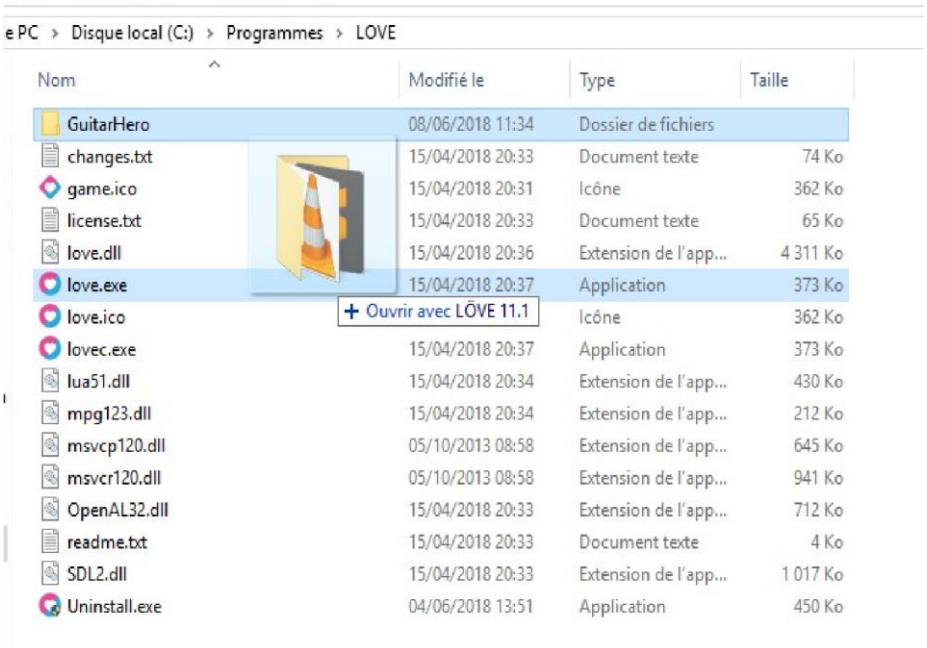
Le jeu est en 2D et demande juste de jouer les bonnes touches qui arriveront de haut en bas au rythme de la musique.

IV. Prise en main de Love2D

L'utilisation de Love2D est assez basique, il suffit de créer, dans votre repository, le fichier « main.lua » avec ce code à l'intérieur :

```
function love.draw()
    love.graphics.setColor(0, 0.66, 0.66, 1)
    love.graphics.rectangle("fill", 300, 300, 50, 30)
end
```

Une fois cela créé, si vous êtes sur Windows, il faudra glisser et déposer le repository sur le fichier exécutable (.exe) de love.



Si vous êtes sur linux, il faudra en revanche taper la commande suivante dans un terminal :

```
love /chemin/vers/votre/dossier/
```

Si vous n'arrivez pas à lancer le projet demandez de l'aide.

V. J'aime quand tout est rangé

Nous allons déclarer des « containers » de variables qui vont stocker des informations utiles.

```
local core = {}
local key = {}
local buf = {}
```

« core » va faire référence à des informations générales (exemple : le menu qu'on doit afficher).

« key » va faire référence aux configurations des notes (exemple : où est-ce qu'elles vont apparaître?).

« buf » est une structure particulière qui va être dynamiquement généré par le moteur du jeu et gère les notes qui apparaîtront à l'écran.

VI. Je vais rajouter les notes

Nous allons maintenant déterminer où est-ce que les notes apparaissent. On va déterminer la largeur et la hauteur ainsi que le nombre de touches différentes (il y a aura 4 touches différentes). Ensuite nous allons configurer la taille de la fenêtre ainsi que de la «scène ».

On configurera également le nombre d'éléments générés par le jeu. Nous faisons ça maintenant en prévision de ce que nous ferons plus tard.

```
function key.setting()
    -- Setup keys config
    key[0], key[1], key[2], key[3] = {}, {}, {}, {}
    key[0].x = 100 key[0].y = -50
    key[0].width = 50 key[0].height = 50
    key[0].mode = 0
    key[1].x = 175
    key[1].y = -50
    key[1].width = 50
    key[1].height = 50
    key[1].mode = 0
    key[2].x = 250
    key[2].y = -50
    key[2].width = 50
    key[2].height = 50
    key[2].mode = 0
    key[3].x = 325
    key[3].y = -50
    key[3].width = 50
    key[3].height = 50
    key[3].mode = 0
end

function love.load()
    success = love.window.setMode(800, 600, flags) core["scene"] = 0
    -- Scene : menu / game / other
    core["touch"] = -1 *
    -- Nbr of items generated by the game
    core["mem"] = 0
    -- Garbage collector cycles
    key.setting()
end
```

A ce niveau-là, vous ne verrez rien graphiquement, mais assurez-vous que tout fonctionne quand vous lancez le jeu !

VII. J'ai mes notes et je créais mes accords

Nous allons créer une fonction pour générer des accords aléatoires. Cette génération d'accords est faite de manière non-optimisé, ça sera à vous de le faire plus tard.

La fonction key.appendBuffer() va générer des objets dans la structure « buf » en se basant sur la configuration des touches qu'on a fait précédemment.

Ensuite, key.scrolling() va passer sur tous les objets générés et changer leurs positions, si un objet est arrivé à la position limite, il est réinitialisé (buf[i] = nil).

La fonction memoryCleaner() va récupérer ces éléments détruits pour les vider de la mémoire utilisée.

Le tout est appelé dans la fonction love.update(dt) qui est une fonction de la librairie.

```
function key.appendBuffer()
    if (math.random(1000) < 25) then
     -- Append number of items.
     core.touch = core.touch + 1
     -- Create object depending on key config.
     buf[core.touch] = {} buf[core.touch].x = key[math.random(4) - 1].x
     buf[core.touch].y = -50
    end
end

function key.scrolling()
    for i = 0, core.touch do
        -- Scrolling objects
        if (buf[i] ~= nil) then
            buf[i].y = buf[i].y + 10
        end
        -- Reset objects
        if (buf[i] ~= nil and buf[i].y == 600) then
            buf[i].y = nil buf[i].x = nil buf[i] = nil
        end
    end
end

function memoryCleaner()
    -- Clean memory
    core.mem = core.mem + 1
    if (core.mem == 500) then
        collectgarbage()
    end
end

function love.update(dt)
    key.appendBuffer()
    key.scrolling()
    memoryCleaner()
end
```

VIII. Je n'ai toujours pas de guitare

Nous allons implémenter la gestion des touches du clavier et introduire le système de score par la même occasion.

Il faudra d'abord rajouter des informations dans « core » puis créer une petite fonction qui va vérifier si la note est touchée ou non et afficher le score.

```
function love.load()
 core["score"] = 0
 love.keyboard.setKeyRepeat(false)
end

function love.draw()
    love.graphics.print("SCORE:", 420, 150, 0, 4, 4)
    love.graphics.print(core.score, 420, 200, 0, 4, 4)
end

function key.checkClicked(x)
    for i = 0, core.touch do
        if (buf[i] ~= nil and buf[i].x == x and buf[i].y > 500 and buf[i].y < 550) then
            buf[i].y = nil
            buf[i].x = nil
            buf[i] = nil
            return (1)
        end
    end
    return (0)
end

function love.keypressed(myKey)
    if myKey == "q" and key.checkClicked(100) == 1 then
        core.score = core.score + 10
    end

    if myKey == "s" and key.checkClicked(175) == 1 then
        core.score = core.score + 10
    end

    if myKey == "d" and key.checkClicked(250) == 1 then
        core.score = core.score + 10
    end

    if myKey == "f" and key.checkClicked(325) == 1 then
        core.score = core.score + 10
    end
end
```

On va également s'occuper du « framerate » :

```
function love.update(dt)
    dt = math.min(dt, 1/60)
end
```

Si vous ne comprenez pas le code n'hésitez pas à poser des questions.

IX. Manque plus qu'un menu

Nous allons élaborer un menu simple avec notamment l'intégration d'une image « logo.png » qu'il faudra mettre dans le dossier du jeu et mettre en mémoire dans le chargement initial du jeu.

```
function love.load()
    core["logo"] = love.graphics.newImage("logo.png")
end
```

Ensuite, nous allons donner du sens à la variable core.scene que nous avons déclaré au tout début.

Une scène correspond à ce qu'on veut montrer au joueur.

Quand il lance le jeu, la scène sera 0, elle correspondra au menu principal. Quand il aura cliqué sur « Jouer » alors la scène changera à 1 et cela correspondra à la scène de jeu.

Cela permet aussi que le programme n'exécute qu'une partie du code suivant la scène.

```
function love.update(dt)
    dt = math.min(dt, 1/60)
    if core.scene == 1 then
        key.appendBuffer()
        key.scrolling()
        memoryCleaner()
    end

    if core.scene == 0 then
        if love.mouse.isDown(1) then
            core.scene = 1
        end
    end
end

function core.drawSceneGame()
    core.ui() for i = 0, core.touch do
        -- If object exists, print it with his right color.
        if (buf[i] ~= nil) then
            love.graphics.setColor(0, 0.66, 0.66, 1)
            love.graphics.rectangle("fill", buf[i].x, buf[i].y, 50, 30)
        end
    end
    love.graphics.setColor(1, 1, 1, 0.5)
    love.graphics.rectangle("fill", 0, 500, 800, 40)
    -- Score part
    love.graphics.print("SCORE:", 420, 150, 0, 4, 4)
    love.graphics.print(core.score, 420, 200, 0, 4, 4)
end

function core.drawSceneMenu()
    love.graphics.setColor(1, 1, 1, 1)
    love.graphics.draw(core.logo, 100, 20)
    love.graphics.setColor(1, 1, 1, 1)
    love.graphics.rectangle("fill", 0, 500, 800, 60)
    love.graphics.setColor(0, 0, 0, 1)
    love.graphics.print("Play", 50, 500, 0, 4, 4)
end

function love.draw()
    if core.scene == 1 then
        core.drawSceneGame()
    end

    if core.scene == 0 then
        core.drawSceneMenu()
    end
end
```

X. J'ai oublié de mettre la musique

A vous de jouer ! Faîtes en sorte que la musique se lance uniquement quand on joue et non quand on lance le jeu. N'oubliez pas de mettre une musique dans le dossier du jeu.

Charger une musique :

```
core["music"] = love.audio.newSource("mama.mp3", 'stream')
```

Lancer une musique :

```
love.audio.play(core.music)
```

XI. Ca manqué d'animation

Nous allons voir une façon de faire une animation dans un jeu.

Il n'y a pas qu'une seule façon de faire, c'est une notion qui est assez complexe car elle demande que vous compreniez bien comment le code s'exécute.

C'est la dernière partie entièrement guidée.

N'oubliez pas d'appeler key.fadeOut() dans love.update(dt)…

```
function key.appendBuffer()
    if (math.random(1000) < 25) then
        buf[core.touch].anim = 1 ( … )
    end
end

function key.checkClicked(x)
    for i = 0, core.touch do
        if (buf[i] ~= nil and buf[i].x == x and buf[i].y > 500 and buf[i].y < 650) then
            buf[i].anim = buf[i].anim - 0.4
            return (1)
        end
    end
    return (0)
end

function key.fadeOut()
    for i = 0, core.touch do
        if (buf[i] ~= nil and buf[i].anim < 1) then
            buf[i].anim = buf[i].anim - 0.1
        end

        if (buf[i] ~= nil and buf[i].anim == 0) then
            buf[i].y = nil buf[i].x = nil buf[i] = nil
        end
    end
end

function core.drawSceneGame()
    for i = 0, core.touch do
        if (buf[i] ~= nil) then
            love.graphics.setColor(0, 0.66, 0.66, 1)
            if buf[i].anim < 1
                then love.graphics.setColor(0, 0.33, 0.33, buf[i].anim)
            end
            love.graphics.rectangle("fill", buf[i].x, buf[i].y, 50, 30)
        end
    end
end
```

XII. Optimiser la création de rectangle

A partir d'ici le sujet vous donnera des pistes pour rendre votre jeu beaucoup plus riche.

Vous n'êtes pas obligé de suivre tous les points du sujet, personnalisez le jeu comme bon vous semble !

Ça sera à vous de le faire à partir de ce qui a déjà été fait jusqu'à présent.

Pour plus d'informations sur la librairie utilisée : https://love2d.org/wiki/love

Pour optimiser la création des rectangles, il faut re-remplir les cases qui ont été mise à 'nil'.

Cela évite que le buffer soit plus grand et que par conséquent le temps de traitement à chaque boucle de jeu soit plus court. Il faudra re parcourir ce qui a déjà été généré, puis quand on tombe sur une case à 'nil' on la remplit. Sinon ont créé une nouvelle case.

XIII. Combos de précision

Ajoutez une variable combo dans « core ». Affichez là grâce à love.graphics.print() dans love.draw().

Maintenant si vous touchez correctement, rajoutez 1 de combo, sinon mettez le combo à 0.

La précision c'est le ratio de "touché" / "essais".

Pour aller plus loin, vous pouvez faire en sorte que la précision soit aussi influencée par le timing (touché trop tôt ou trop tard) ou prendre en compte les touches qui quittent l'écran.

XIV. Niveaux de difficultés

Vous pouvez créer un menu pour choisir la difficulté de jeu ou bien augmenter la difficulté au fur et à mesure par exemple.

Pour augmenter la difficulté, vous pouvez augmenter la probabilité de générer un rectangle dans key.appendBuffer() ou bien accélérer l'arrivée des touches.

XV. Animation touche enfoncée

Ce serait plus sympa si on voyait visuellement qu'on appuie sur une touche. La difficulté de cette partie c'est de faire une animation identique pour les 4 touches, qu'on puisse en faire 4 simultanément et que, si la touche est ré-appuyée pendant l'animation, elle relance l'animation ou qu'elle en relance une sur celle qui est en cours.

Quelques indications pour faire une animation qui réinitialise une animation en cours si la touche est ré-appuyée : il faudra rajouter ceci dans l'initialisation du jeu.

```
core['animation'] = {}
core['animation'].q,    core['animation'].s,    core['animation'].d,
core['animation'].f = 0, 0, 0, 0
```

L'objectif maintenant est de faire passer l'état de q,s,d,f à un nombre supérieur à 0 quand la touche est appuyée. Ce nombre correspondra au nombre d'étapes dans l'animation.

On peut imaginer maintenant qu'on affichera un rectangle sur les zones où les touches doivent arriver et qui rétrécit au fur et à mesure par exemple.

XVI. Score avancé

Rajoutez plus ou moins de score en fonction des combos. Utilisez des images pour afficher le score : une image pour le 0, 1, 2 …

```
core["logo"] = love.graphics.newImage("logo.png")
love.graphics.draw(core.logo, 100, 20)
```

XVII. Game Over

En parcourant la documentation, vous devriez être en mesure de créer un timer basé sur le temps (et non pas le framerate) pour créer un temps limite de jeu (fin de la musique par exemple).

Ensuite vous devrez créer un petit écran de 'Game Over' et revenir au menu.

ATTENTION!

Si le menu de 'Game Over' revient sur le menu, il faudra bien remettre à zéro tous les paramètres du jeu comme si on avait lancé le jeu pour la première fois! Cela évitera des conflits lorsqu'on relancera une partie.

XVIII. Génération avancée des rectangles

Vous pouvez vous inspirer de 'Subway Surfer' qui est un jeu de plateforme où les obstacles sont générés de manière aléatoire avec des motifs prédéfinis.

L'idée ici serait donc de créer des 'motifs' dans le code puis appeler la génération du motif aléatoirement avec un certain délai pour qu'un ‘motif’ n’écrase pas un autre.

XIX. L’heure des bonus

- Affichage des rectangles en perspective

Munissez-vous de la documentation et de Google pour chercher ce que sont les 'RayTracer'. Sinon, prenez des images pour les touches et définissez une trajectoire et une échelle appropriée pour donner un effet de perspective.

- Intelligence Artificielle

Joue au jeu à votre place ou jouez contre..

- UI Complexe

Créer un vrai design de jeu. Faire des boutons avec un effet 'survol'. Utiliser des typographies personnalisées. Fond de jeu animé. Highscores affichés. Notes avec un style particulier...

- Créateur de cartes

Créer des maps directement dans le jeu grâce à un éditeur qui enregistrera sous forme de fichier texte. Sinon 'parser' un fichier fait sur un éditeur de texte et générer les touches en fonction de ce fichier.

Cela vous demandera de regarder plus en profondeur les fonctions systèmes du Lua. Il faudra sûrement accompagner ce système d'un menu de sélection de cartes.

- Mode Versus en local

Mode écran scindé avec une partie du clavier pour joueur 1 et 2 ou mode LAN.

- Menu Options

Modifier le volume. Changer le thème visuel du jeu. Changer les images par seconde…

- Gameplay amélioré

Créer une barre de vie qui se vide en permanence et qui se remplit dès qu'on touche correctement. Si la barre atteint 0, vous perdez.