Soutenance Module 1 – Top Down Shooter
Karim Gouyette – Promotion Casablanca
(15/12/2024)

1. Comment jouer ?

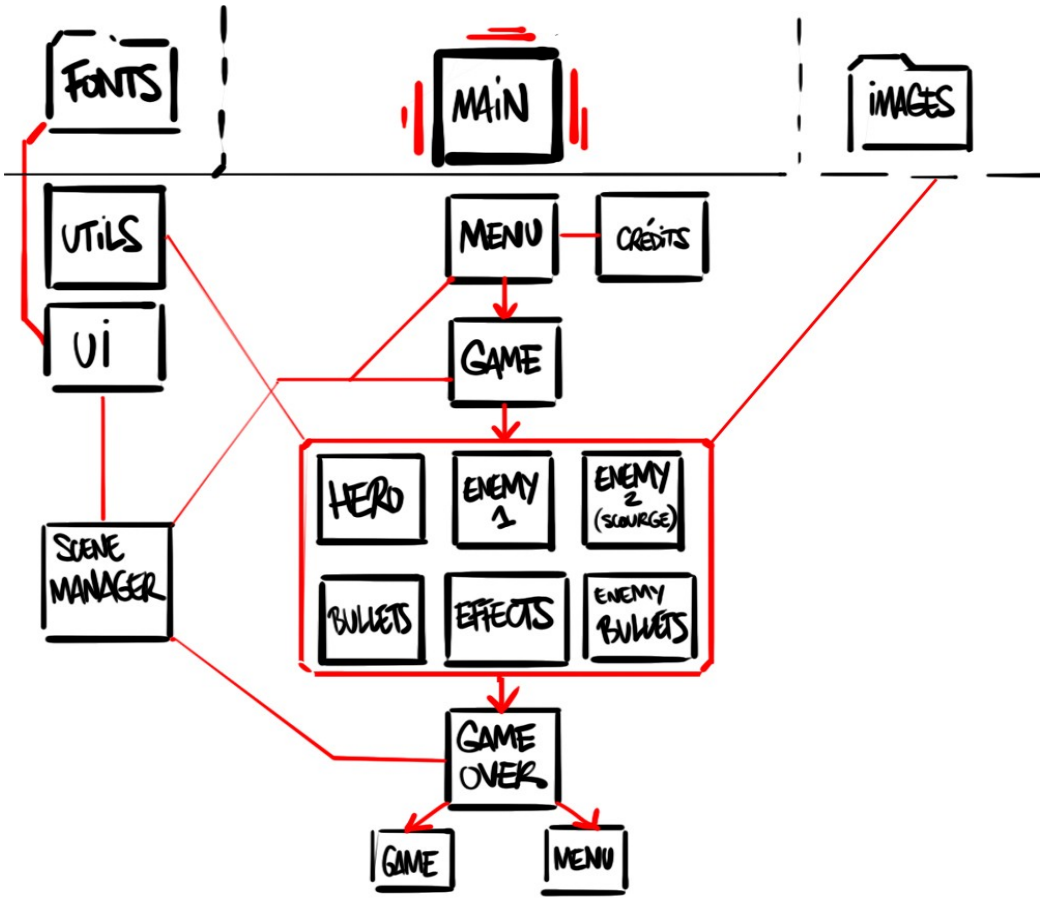
Dans *XSHTX*, **l'objectif** est de gagner un maximum de points en éliminant des ennemis tout en survivant le plus longtemps possible à leurs assauts illimités.

Commandes:
Clic-droit pour tirer, *pointer* pour viser et *WASD/ZQSD* pour se déplacer.

On **gagne** en restant le plus longtemps en vie, il suffit d'une collision ou tir ennemi pour **perdre** instantanément et enregistrer l'accumulation de points.

2. Code source.

Structure :



Découpage :

Le cœur du projet s'articule autour du déplacement et de la collision avec effets visuels entre les entités à l'intérieur des 6 modules encadrés en rouge (héros, 2 types d'ennemis, deux types de projectiles et leurs effets visuels)

Ces modules sont pour la plupart découpés en tables qui abritent des fonctions sur le modèle suivant :

```
function createEnemy()
   local enemy = {}
   local spawnRadius = 600
   enemy.x, enemy.y =  10, 10
   enemy.life = 10
        etc.
   enemy.update = function(dt)
   end
   enemy.chargeState = function(dt)
   end
   enemy.checkHeroDistance = function()
   end
        etc.
```

Pour les déplacements et l'orientation, chacun des ennemis, héros et projectiles se partagent les formules clés suivantes à l'intérieur de leurs modules respectifs :

```
-- Returns the distance between two points.
function math.dist(x1,y1, x2,y2) return ((x2-x1)^2+(y2-y1)^2)^0.5 end

-- Returns the angle between two vectors assuming the same origin.
function math.angle(x1,y1, x2,y2) return math.atan2(y2-y1, x2-x1) end
```

Jusque dans les fonctions de collisions :

```
function isIntersecting(x1, y1, r1, x2, y2, r2 )
   local dist = math.dist(x1,y1,x2,y2)
   local radiusSum = r1 + r2
   return dist < radiusSum
end
```

Enfin pour renforcer l'aspect visuel et sensoriel (« Game feel »), les particules et secousses sont intégrées aux collisions et tirs de façon aléatoire :

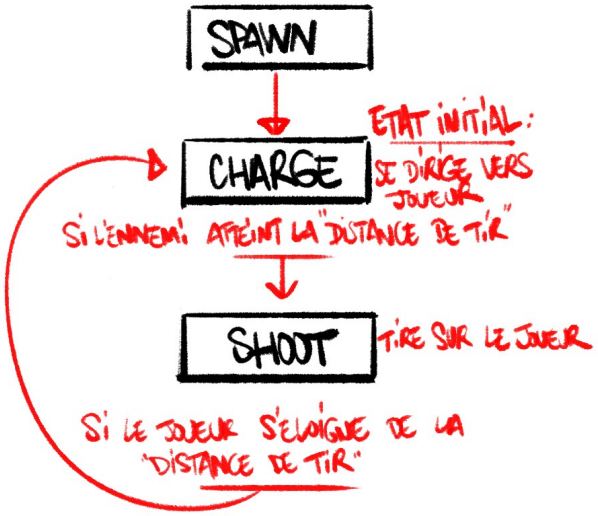
```
function shakeDraw()
   if t < shakeDuration then
      local dx = love.math.random(-shakeMagnitude, shakeMagnitude)
      local dy = love.math.random(-shakeMagnitude, shakeMagnitude)
      love.graphics.translate(dx, dy)
   end
end
```

```lua
function ajouteBlast(pX, pY)
   local monBlast = {}
   monBlast.x = pX
   monBlast.y = pY
   monBlast.r = love.math.random(5,20)
   monBlast.vx = love.math.random(-400,400)/200
   monBlast.vy = love.math.random(-400,400)/200
   monBlast.vie = love.math.random(50,400)/1000
   monBlast.c = {0,0,0,0}
   monBlast.speed = 30
   table.insert(listeBlasts, monBlast)
end

function updateBlast(dt)
   for n=#listeBlasts, 1, -1 do
      local blast = listeBlasts[n]
      blast.x = blast.x + blast.vx * (blast.speed * dt)
      blast.y = blast.y + blast.vy * (blast.speed * dt)
      blast.vie = blast.vie - dt
      if blast.vie <= 0 then
         table.remove(listeBlasts, n)
      end
   end
end
```

Machine à état :

Elle est utilisée pour le deuxième ennemi (enemyScourge.lua) très simplement pour le faire tirer à une distance prédéfinie par le joueur après son apparition en s'arrêtant, puis de se re-déplacer dans sa direction jusqu'à atteindre à nouveau la distance prédéfinie si le joueur s'éloigne.

```lua
scourge.shootState = function(dt)
      local speed = 1
      scourge.angle = math.atan2(hero.y - scourge.y, hero.x -
scourge.x)
      scourge.x = scourge.x + math.cos(scourge.angle) * speed * dt
      scourge.y = scourge.y + math.sin(scourge.angle) * speed * dt
      scourge.shoot()


      if math.dist(hero.x, hero.y, scourge.x, scourge.y) >
hero.radius + scourge.shootRange then
         scourge.state = scourge.chargeState
      end
   end
```



```
scourge.chargeState = function(dt)
      scourge.angle = math.atan2(hero.y - scourge.y, hero.x - scourge.x)
      scourge.x = scourge.x + math.cos(scourge.angle) * scourge.speed * dt
      scourge.y = scourge.y + math.sin(scourge.angle) * scourge.speed * dt
      scourge.checkHeroDistance()

      if math.dist(hero.x, hero.y, scourge.x, scourge.y) < hero.radius + scourge.shootRange and
      scourge.x - scourge.width * 2 > 0 and scourge.x - scourge.width * 2 < SCR_WIDTH and
      scourge.y - scourge.height * 2 > 0 and scourge.y + scourge.height * 2 < SCR_HEIGHT then
         scourge.state = scourge.shootState
      end
   end
```

Notes :

Inspirations : *Hotline Miami*, *Nuclear Throne* et *Tormentor X Punisher* pour l'intensité du « game feel ».

Features désirées: rendu en basse résolution et grossièrement pixelisé, développement de la direction des particules de sang avec persistance sur le sol, ouverture du canvas, boss et clic-gauche pour autre arme (ou recharge).