Proyecto final

Evelyn Tabares Valencia

Asignatura:
Computación gráfica

Docente:
Andrés Felipe Ramírez Correa

Ingeniería de software y computación
Universidad tecnológica de Pereira
2022

Funcionamiento del software

Enlace del juego

https://proyecto-final-computacion-grafica.evelyntabaresv.repl.co/

Nota: *Abrir preferiblemente en Google Chrome*

Enlace de archivos online:

https://replit.com/@evelynTabaresV/Proyecto-final-computacion-grafica#code/main.js

Interfaz gráfica de usuario:

La primera imagen que el usuario encuentra es la de **Inicio.** Esta al igual que las siguientes, fue creada en la página https://www.canva.com/ Con el fin de otorgarle al jugador una interfaz amigable y que estuviera acorde a la temática del juego.

Al pulsar Enter ingresa a la siguiente interfaz.

Interfaz de **Controles:** Le da al usuario una guía de los controles de juego y su objetivo.

Interfaz de **Créditos:**

Aparece al final del juego.

Interfaz de **Ganador**

Interfaz **Perdedor**

Al pulsar Enter ingresa a la siguiente interfaz.

La interfaz de **Juego:**

Está constituido de la siguiente manera:

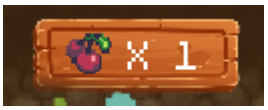
Vidas y puntaje

Al lado izquierdo de la pantalla encontramos una tabla, nos indica la cantidad de vidas actuales y el puntaje obtenido, el cual está definido de la siguiente manera:

score: puntaje actual **/** puntaje objetivo.

El **puntaje objetivo**: cambia según el nivel, dicho valor al ser alcanzado o sobrepasado por el jugador, le permite cruzar la puerta y por ende pasar de nivel.

Cerezas



Las cerezas se encuentran distribuidas por toda la pantalla. Una vez el jugador recolecta **20** de estas, **recupera una vida.**

Cofres

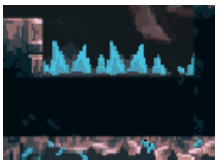
Los cofres pueden encontrarse en lugares específicos en cada nivel. El jugador puede abrirlos al pulsar Z y obtendrá un diamante.

Los diamantes son bonos de puntaje, tienen un valor aleatorio de: [5, 10, 15 ,25].

Plataforma flotante

La plataforma flotante sube y baja de manera vertical. Al subirse el jugador en esta, se detiene, por lo tanto, si lo que desea es subir debe saltar para que siga moviéndose.

Picos azules




Se encuentran en el último nivel, si el personaje cae sobre estos o los toca, muere instantáneamente.

Enemigos

Gato: El gato es el enemigo más débil del juego, para derrotarlo el jugador solo debe golpearlo una vez, al colisionar con el usuario le descontarán **2** puntos de vida. (El puntaje total de vida es **4500**, cada corazón equivale a **1500** puntos). 

```
action('gato', (g) => {//Acción del judador al tocar el gato
  if (g.muerto == false) {//Si el gato está vivo
    //Si el jugador toca el gato sin estar pulsando X (ataque)
    if (player.isTouching(g) && !(isKeyDown("x"))) {
      lives -= 2//Puntos de vida descontados por cada toque
    }
  }
})
```

Ataque al enemigo

```
collides('espada', 'gato', (k, g) => {//Si el enemigo es golpeado por la espada
  if (g.muerto == true) return;
  shake(2)//movimiento de la camara
  wait(1, () => {
    destroy(k)// destruye el efecto de golpe de la espada
  })
  g.squash(); //muerte del gato
  score += 10//aumenta el puntaje en 10 por cada gato
})
```

Si el gato es golpeado por la espada, siempre y cuando esté vivo, la cámara del juego se sacudirá y después de 1 segundo se destruye el efecto de la espada. Luego de esto el código llamará a una función llamada ‘squash’, ubicada dentro de la función ‘enemy’.

La función ‘squash’, me permite matar al enemigo y reproducir un sonido al destruirlo, a la par de una animación.

Al destruir al enemigo sumamos al puntaje actual **10 puntos**. Cada gato equivale a dicha cantidad.

```
//-Función de definición de enemigos / contiene acción de muerte y daño al enemigo
function enemy() {
  return {
    id: "enemy",
    require: ["pos", "area", "sprite", "patrol"],
    muerto: false,
    update() { },
    squash() {
      console.log('squashing');
      this.muerto = true;//Se indica que el enemigo está muerto
      this.unuse("patrol");//Se deja de usar la función 'Patrol'-> Caminata
      this.stop();
      wait(0.6, () => {//Esperar 0.6 segundos para reproducir
        this.play("muerte")//Animación de muerte del enemigo
      })
      play('death')//Sonido muerte
      this.use(lifespan(2, { fade: 1 }));//Tiempo de muerte del enemigo 2sec y se desvanece durante 1 sec

      // destroy(this)
    },
    dañoE() { //daño al esqueleto
      this.stop()  //Se detiene la caminata
      if (this.curAnim() !== 'daño') { //se cambia la animación actual por 'daño'
        this.play('daño');//Animación del enemigo recibiendo daño
      }
```

Esqueleto:

Este es el enemigo del **nivel 2**, se debe golpear más veces para derrotarlo, se necesita específicamente 6 golpes. Con cada colisión de la espada con el enemigo, se sumará un golpe y luego se ejecuta la función ‘dañoE’, ubicada dentro de la función ‘enemy’.

```
dañoE() { //daño al esqueleto
  this.stop()  //Se detiene la caminata
  if (this.curAnim() !== 'daño') { //se cambia la animación actual por 'daño'
    this.play('daño');//Animación del enemigo recibiendo daño
  }
```

Esto detiene el movimiento del enemigo y si la animación que ejecuta en ese momento es diferente a la de recibir daño, entonces la ejecuta. Al llegar a propiciar seis golpes se ejecuta la función ‘squash’, mencionada en la explicación anterior del enemigo ‘gato’. Cada esqueleto equivale a **25 puntos** y se reinician los golpes para el siguiente esqueleto.

```
collides('espada', 'esqueleto', (k, s) => {//Si el enemigo es golpeado por la espada
  if (s.muerto == true) return;//Si está muerto no debe ejecutar ninguna acción y retornar
  shake(2)//movimiento de la camara
  wait(1, () => {//Esperar 1 segundo para
    destroy(k)// destruye el efecto de golpe de la espada
  })
  golpes += 1//contador de veces que el jugador toca al esqueleto
  s.dañoE()//animación daño al esqueleto

  if (golpes == 6) {//Se necesitan 6 golpes al esqueleto para matarlo
    s.squash(); //muerte del esqueleto
    score += 25//aumenta el puntaje en 25 por cada esqueleto
    golpes = 0//se inicializa en 0 para el siguiente enemigo
  }
})
```

Condiciones para disminuir vida del jugador

Si el jugador colisiona con el esqueleto y no está pulsando X para atacar, su vida disminuye en **0.5 puntos** en cada toque.
En este momento el esqueleto ejecuta una animación de ataque hacia el jugador, espera 0.7 segundos y vuelve a caminar.

Gusano:

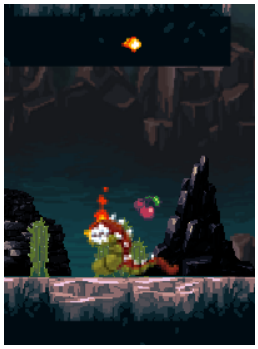
Este es el enemigo del nivel final y está definido a través de 5 estados: "idle", "attack", "walk", "gethit", "death", a través de los cuales hace un ciclo hasta llegar al estado ‘death’.

```
//-------------------------ENEMIGOS-------------------------------
const SPEED = 320
const ENEMY_SPEED = 160
const BULLET_SPEED = 700//800
let vivo = true// gusano está vivo
if (levelId == 2) {//Enemigo final solo se ejecuta en dicho nivel
  const enemyfin = add([
    sprite("gusano", { anim: 'idle' }),
    pos(400, 400),
    origin("center"),
    body(),
    solid(),
    scale(1.5),
    area(),
    "gusano",
    // This enemy cycle between 3 states, and start from "idle" state
    state("walk", ["idle", "attack", "walk", "gethit", "death"]),
  ])
```

El ciclo inicia en el **estado 'idle'**, donde se ejecuta la animación de respiración, espera 0.5 segundos y se pasa al estado de 'attack'.

```
  enemyfin.onStateEnter("idle", async () => {
    enemyfin.play("idle") //Animación  respiración
    await wait(0.5)
    enemyfin.enterState("attack")
  })
```

En el **estado 'attack'** se crea una bola de fuego, si el jugador existe y el gusano está vivo. El gusano ejecuta la animación de ataque y se crea la bola de fuego según la posición del gusano. Su movimiento en cambio está dado por la dirección en la que se encuentre el jugador. Haciendo que la bola salga disparada hacia este.



Luego de esperar 0.7 segundos para ejecutar la siguiente acción, entra al estado ‘walk’.

```
enemyfin.onStateEnter("attack", async () => {//Estado de ataque
  // Don't do anything if player doesn't exist anymore
  if (player.exists() && vivo == true) {//Si el jugador existe y el gusano está vivo
    enemyfin.play("attack") //Animación  ataque
    const dir = player.pos.sub(enemyfin.pos).unit()// Toma el valor de la dirección del gusano
    add([ //Se agrega una bola de fuego
      pos(enemyfin.pos),//La posición de la bola de fuego es la misma del gusano
      move(dir, BULLET_SPEED),//Su dirección es la del jugador
      sprite('bolafuego', { anim: 'bola' }),//Sprite de la bola de fuego
      area(),
      cleanup(),
      origin("bot"),
      "bullet",
    ])
  }
  await wait(0.7)//espera antes de ejecutar la siguiente acción
  enemyfin.enterState("walk")//Entra al estado caminata
})
```

En el **estado ‘walk’**, se ejecuta la animación de caminata, se espera 2 segundos y se entra al estado ‘idle’.

```
enemyfin.onStateEnter("walk", async () => {
    enemyfin.play("walk")//Animación caminata
    await wait(2)
    enemyfin.enterState("idle")

})
```

Para el estado ‘death’, se espera 0.5 segundos, se ejecuta la animación de muerte y se destruye al enemigo.

```
enemyfin.onStateEnter("death", async () => {
  await wait(0.5)
  wait(1, () => {
    enemyfin.play("death") //Animación muerte
    destroy(enemyfin)//Se destruye al gusano
  })
  vivo = false//Se define como falso, el gusano ha muerto
  play('death')//Sonido muerte
  destroy(bullet)//Se destruye la bala
  enemyfin.use(lifespan(2, { fade: 1 }));//se desvanece durante 1 segundo

})
```

En el **estado ‘gethit’** se ejecuta la animación de recibir daño, se espera 0.5 segundos y se regresa al estado ‘idle’.

```
enemyfin.onStateEnter("gethit", async () => {
  enemyfin.play("gethit") //Animación recibir daño
  await wait(0.5)
  enemyfin.enterState("idle")//Estado de respiración
})
```

Ataque al enemigo:

El gusano con cada golpe entra al estado ‘gethit’ (recibir daño). Al recibir el daño necesario, pasa al estado ‘death’ y al destruir al enemigo sumamos al puntaje actual **50 puntos**. Cada gusano equivale a dicha cantidad.

```
collides('espada', 'gusano', (k, u) => {//Si el enemigo es golpeado por la espada
  shake(2)//movimiento de la camara
  wait(1, () => {
    destroy(k)// destruye el efecto de golpe de la espada
  })
  golpes += 1//Aumenta la cantidad de golpes en cada toque

  enemyfin.enterState("gethit")//Cambio de estado del gusano/pasa a recibir daño
  if (golpes == 6) {//Al llegar a propiciarle 6 golpes
    enemyfin.enterState("death")// Cambio de estado a 'muerte'
    score += 50//aumenta el puntaje en 50 por cada gusano
  }
```

Código del software

Se define un fondo de pantalla y puntaje objetivo para cada nivel

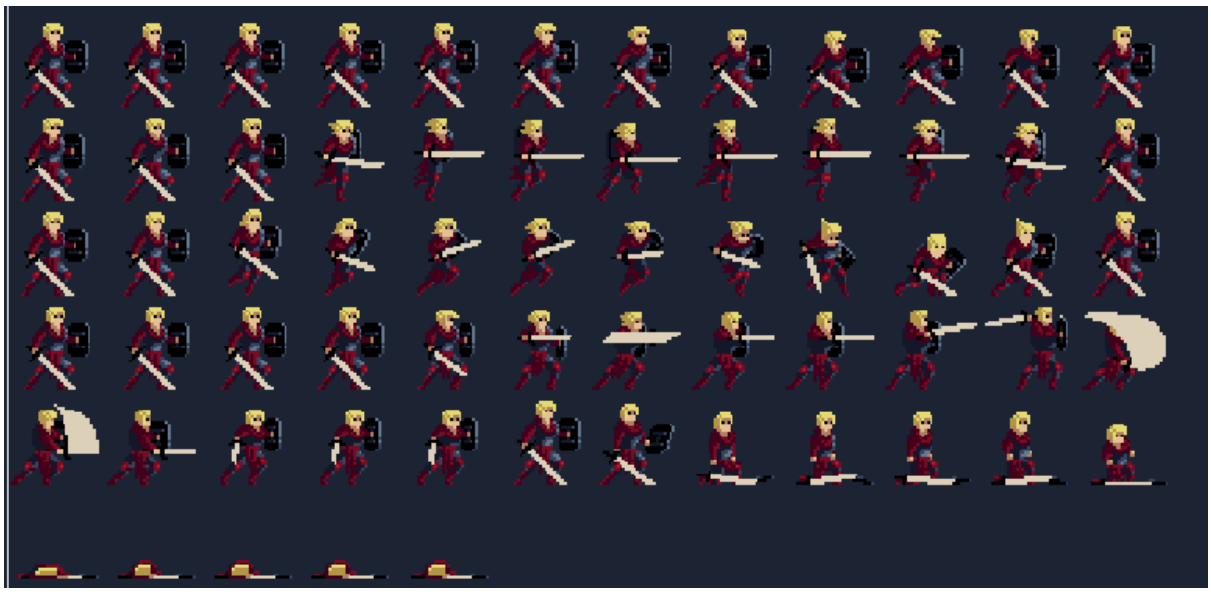
```
//Cambio de Fondo Según el Nivel
if (levelId == 0) {
  add([sprite('bg'), layer('bg'), area(), { width: width(), height: height() }])//Fondo nivel1
  objetivo=30 //Puntaje objetivo nivel 1
}
else if (levelId == 1) {
  add([sprite('bg2'), layer('bg'), area(), { width: width(), height: height() }])//Fondo nivel2
  objetivo=65 //Puntaje objetivo nivel 2
}
else if (levelId == 2) {
  add([sprite('bg3'), layer('bg'), area(), { width: width(), height: height() }])//Fondo nivel3
  objetivo = 110 //Puntaje objetivo nivel 3
}
```

Se define el **objeto jugador:** posición, escala, tipo de objeto y sprite

```
//Define al objeto 'jugador'
const player = add([
  sprite('Atlas_jugador'),//'Atlas_jugador'
  pos(240, 40),//95 20
  scale(1.3),
  area(),
  body(),
])
player.play("quieto")//Animación de respiración del personaje
//Define al enemigo 'esqueleto'
```

El Sprite del jugador ‘Atlas_jugador’ contiene cada acción del personaje, ejecutadas según las decisiones del usuario.

```
//Sprites del jugador
loadSprite('Atlas_jugador', 'sprites/Atlas_jugador.png', {
  sliceX: 12,//Número de imagenes en X
  sliceY: 6,//Número de imagenes en Y
  "x": 421,//Tamaño en X
  "y": 205,//Tamaño en Y
  anims: { //Animaciones de cada acción
    quieto: { from: 0, to: 13, loop: true, speed: 5 },
    correr: { from: 15, to: 22, loop: true, speed: 10 },
    salto: { from: 24, to: 36, loop: true, speed: 29 },
    ataque: { from: 37, to: 52, loop: false, speed: 40 },
    muerte: { from: 53, to: 64, loop: true, speed: 5 }
  },
})
```



Siguiendo la lógica anterior se hizo lo mismo para cada personaje incluido en el juego: Enemigos, diamante, cofre y cerezas.

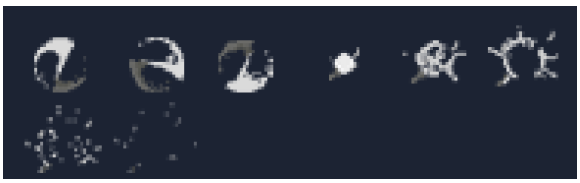
Acción ataque

Al oprimir X en el teclado para atacar, se ejecutará la animación de ataque del jugador, siempre y cuando no se esté ejecutando en ese momento.

```
onKeyDown("x", () => {// Acción de ataque
  if (player.curAnim() !== "ataque") { //Si la animación actual no es 'ataque'
    player.play("ataque")
    if (right == false) {//El jugador está volteado a la izquierda
      spawnKaboom(player.pos.sub(5, -17))//la posición del golpe de espada cambia
    }//cuando el jugador está volteado a la derecha
    else { spawnKaboom(player.pos.sub(-55, -17)) }
  }

})
```

Si el jugador está mirando a la izquierda o a la derecha, se posiciona un sprite de efecto de golpe de espada a ese lado. Este en realidad un objeto, el cual estará en contacto con el enemigo, nos ayuda a saber en qué momento se ataca y a disminuir puntos cuando la ‘espada’ no toque al enemigo.



Efecto de golpe de espada: en el código se puede distinguir como ‘espada’.

Creación de la espada: Su posición es la del jugador y se destruye luego de 1 segundo.

```
function spawnKaboom(p) {//Aparece un efecto de golpe de espada en la posición 'p'
  const obj = add([sprite("espada", { anim: 'espadaG' }), area(), pos(p), 'espada',
  rotate(player.angle),
  origin("center")])
  wait(1, () => {//Espera 1 segundo
    destroy(obj)// Destruye el efecto
  })
}
```

Acción del cofre

Al abrir un cofre, este desaparece, dejando un diamante, el cual para recolectar debe pulsar igualmente la tecla Z. Se puede obtener un diamante con un valor random entre [5, 10, 15, 25]. Dicho valor se suma al puntaje como un **bonus de puntos.**

```
//Define la interacción del jugador con el cofre
onKeyPress("z", () => {// Al pulsar 'z' en el teclado se abre el cofre/ recolecta el diamante
  every("cofreabierto", (c) => {
    if (player.isTouching(c)) { //Si el jugador toca el cofre
      if (c.opened == false) {//Si el cofre está cerrado
        play('chest')//sonido cofre abierto
        c.play("abierto")//Animación cofre abierto
        if (!hasDiamante) {//Si es la primera vez que entra al condicional
          //Crear diamante
          const diamante1 = level.spawn("D", c.gridPos.sub(-0.5, -0.5))/// aparece en la misma posicion que el cofre
          hasDiamante = true//el diamante existe
          wait(0.5, () => { destroy(c) })//Esperar 0.5 segundos para destruir el cofre
        }
      }
    }
  })
  if (hasDiamante) {//Si ya está creado el diamante----Recolección de diamante
    every("diamante", (d) => {
      if (player.isTouching(d)) {//si el jugador toca el diamante y oprime 'z'
        play('recolectar')//Sonido de recolectar
        wait(0.3, () => {//Espera 0.5 segundos
          destroy(d) //destruye el diamante
        })
        hasDiamante = false//Se inicializa en true para los siguientes cofres
        //la función 'Choose' escoge uno de los puntajes de forma aleatoria
        const num = choose([5, 10, 15, 25])//Posibles diamantes
        score += num// Se suma dicho puntaje al score general
      }
```

Cereza

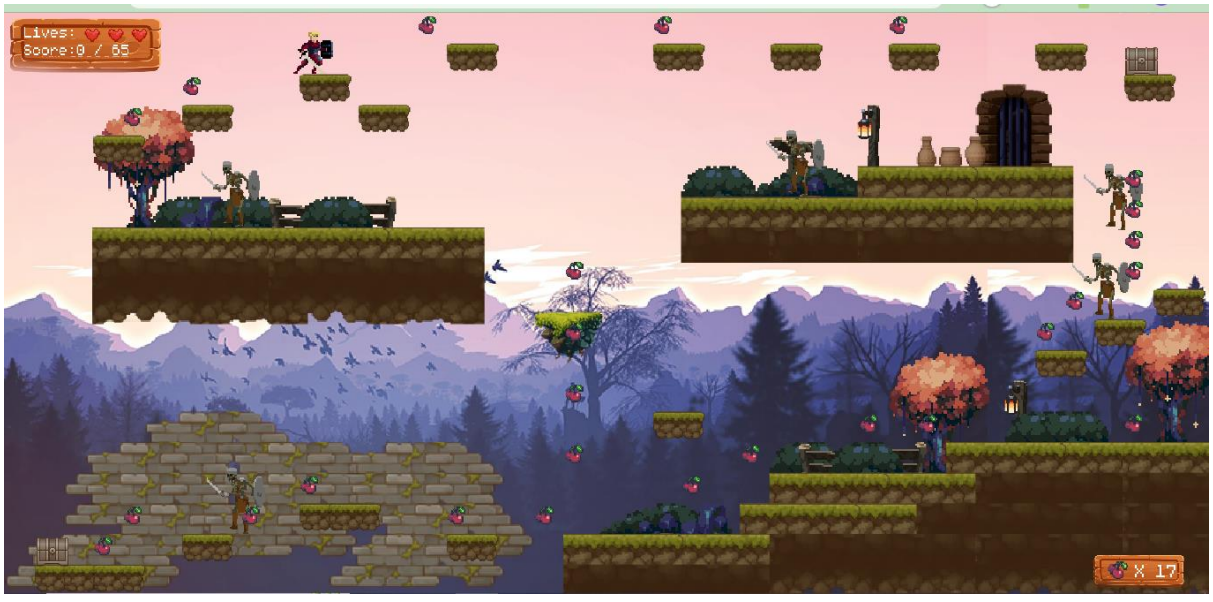
Al tocar una cereza se destruye y se suma al total. Al recolectar 20 se recupera **1 vida.**

```
player.onCollide("cherry", (c) => {//Recolección de cerezas
  destroy(c)//Al tocar una cereza, se destruye
  play("recolectar")//Sonido de recolección
  cereza += 1//Se suma 1 cereza al usuario
  if (cereza == 20) {//Al recolectar 20 cerezas obtiene una vida
    lives += 1500//Equivale a 1 vida
    cereza = 0//se reinicializa
  }
```

Vista previa

Nivel 1

Nivel 2



Nivel 3