D0DTCH!

Videojuego realizado en pygame

Realizado por David Bernal Navarrete

Tabla de contenido

El juego

Concepto

El juego trata de esquivar proyectiles generados con una posición y una dirección aleatorias.

El jugador debe moverse para evitar entrar en contacto con dichos proyectiles, y así sobrevivir la máxima cantidad de tiempo.

La vida y energía del jugador son limitadas: la vida se pierde al recibir golpes; y la energía se pierde al moverse. Ambos valores comienzan y están limitados a 100.

Dificultades

El juego tiene 5 dificultades, dos de las cuales (la más fácil y la más difícil) no están a la vista del usuario de forma directa.

La dificultad cambia la velocidad del proyectil, la velocidad de movimiento del jugador, y la vida perdida por recibir un impacto.

Las dificultades son las siguientes:

1. Baby Mode: La dificultad más fácil. El jugador se puede mover a la máxima velocidad, los proyectiles avanzan a la mínima velocidad, y se pierde 1 de vida al recibir un impacto. Para seleccionarla el usuario debe intentar cambiar a una dificultad más fácil 5 veces, estando ya seleccionada “Fácil”.
2. Fácil: La más fácil directamente visible al jugador. Los proyectiles son lentos, el jugador rápido, y puede recibir hasta 10 golpes antes de perder.
3. Normal: La dificultad inicial. Los proyectiles dan tiempo al jugador a responder, y tiene la velocidad suficiente para, si no reacciona a tiempo, ser golpeado. Se pueden recibir hasta 5 golpes en esta dificultad.
4. Difícil: La más difícil directamente visible al jugador. Los proyectiles aparecen más rápido, y si no se reacciona en el momento en el que salen, no se pueden esquivar. Se pueden recibir hasta 3 golpes antes de perder.
5. Bullet Hell: La dificultad más difícil. Los proyectiles aparecen y desaparecen, y con la velocidad del personaje, aunque se reaccione a tiempo, si la trayectoria coincide con el jugador de forma directa, no se pueden esquivar. Se pueden recibir hasta 2 golpes. Para

seleccionarla, el jugador tiene que intentar subir la dificultad 5 veces estando ya seleccionada “Difícil”.

Controles

Los controles son los siguientes:

- Flechas de dirección: Controlan el movimiento entre botones en los menús y el movimiento del jugador.
- Z: Actuar.
- X: Cancelar/Atrás.
- Escape: Pausa.
- Q: Mata al jugador.

Puntuación

La puntuación consiste en la cantidad de segundos que el jugador ha aguantado vivo, junto con la dificultad en la que se ha jugado. A mayor tiempo sobrevivido, mayor será la puntuación, y a mayor la dificultad, mayor el valor sentimental de dicha puntuación.

Sonido

La música del juego (tanto la de la pantalla de muerte como la del juego) han sido generadas de forma aleatoria usando una red neuronal en Python, cuyo autor es CodeParade, y cuyo código se puede encontrar aquí.

En cuanto a los efectos de sonido, se han usado dos:

- El sonido del golpe es este.
- El sonido de la flecha posee licencia de Atribución 3.0, y es un sonido producido por Mike Koenig.

Gráficos

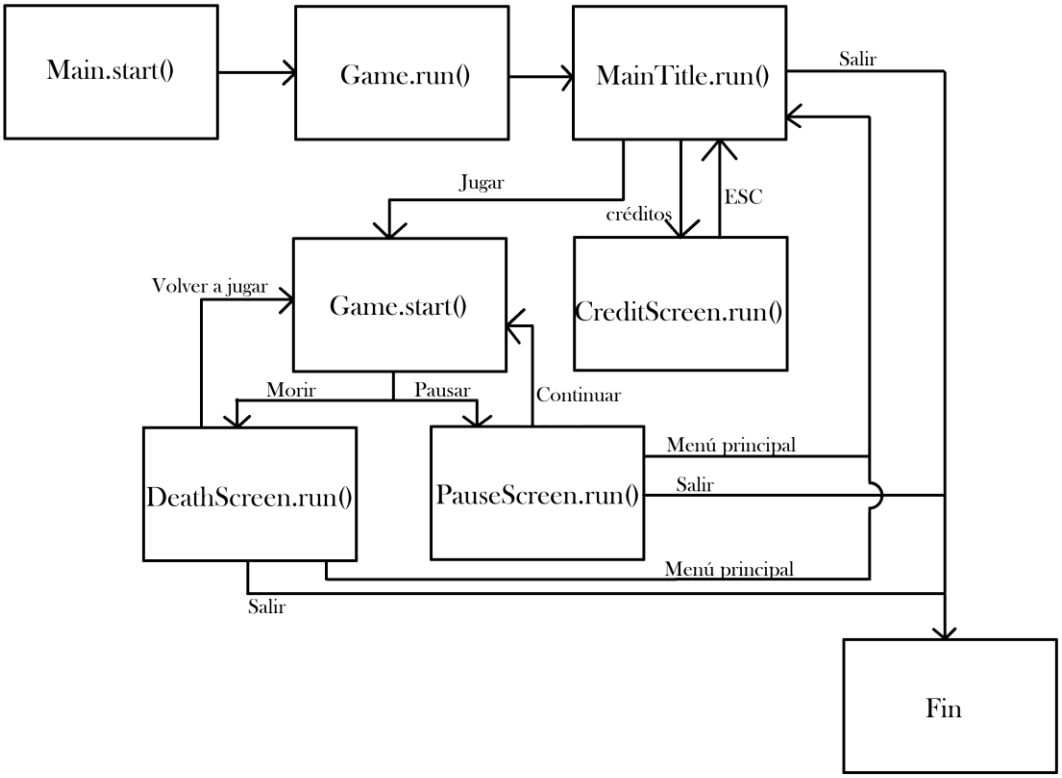
Tanto la imagen de fondo como los sprites para los proyectiles y el jugador han sido diseñados por el desarrollador.

El código

Todo el código menos los comentarios están escritos en inglés por inercia del autor, y no han sido traducidos al español porque al autor le parece más elegante así.

El bucle del programa

Se navegará entre distintos archivos .py, para separar y diferenciar cada etapa del juego. Para controlar la navegación se usarán dos variables: *playState* y *playAgain*. La última se usará para controlar cuándo se ha seleccionado volver a jugar al perder una partida; la otra se usa para saber si se continúa, se vuelve al menú principal, o se sale del programa.



Archivo Config.py

En este archivo se encuentran las variables, funciones, y objetos globales.

Las variables se utilizan para controlar el funcionamiento del juego, el funcionamiento del audio, almacenar la puntuación, la dificultad escogida, las localizaciones de imágenes y archivos de audio, colores, altura y anchura del monitor, y diversos objetos *pygame.Rect* para los botones y límites de movimiento.

Las funciones se usan para dibujar botones, incrementar y decrementar la dificultad, generar proyectiles, y comprobar los límites establecidos para el movimiento.

Las clases son las necesarias para los objetos *Player* y *Projectile*.

Variables

A continuación, una lista de las variables con una breve descripción de su uso y necesidad:

Variables para el funcionamiento general del programa

1. running: Bool para controlar el funcionamiento general del programa. Si es false, no se ejecutará nada.
2. audio: Bool que controla el funcionamiento del audio. Si es True, se reproducirá audio y SFX.
3. gameRunning: Bool que controla el funcionamiento del **juego**, es decir, cuando el programa se encuentra en *Game.start()*.
4. clock: Objeto de tipo *pygame.time.Clock()*. Sirve para controlar la velocidad a la que se ejecuta el programa.

5. points: Entero para almacenar la puntuación del jugador.

Variables para la dificultad del juego

1. gameDificulty: Entero para el nivel de dificultad. En orden ascendente de dificultad, desde 0 hasta 4 (ambos inclusive): Baby Mode, Fácil, Normal, Difícil, Bullet Hell.
2. gameDifficultyName: Cadena para almacenar el valor que se mostrará en el selector de dificultades. Su valor se cambiará en *Config.assignDifficultyName()*.
3. babyModeCounter y bulletHellCounter: Enteros contadores para las dificultades especiales. Se usarán para saber si se ha llegado a intentar subir/bajar de dificultad 5 veces seguidas.

Variables para las rutas de los archivos

Imágenes

1. bgMainTitle: Imagen de fondo para el título principal.
2. bgGame: Imagen de fondo para el juego.
3. gameIcon: Icono para la ventana.
4. bgPauseScreen: Imagen de fondo para la pantalla de pausa.
5. playerStill, playerMoveLeft, playerMoveRight, playerHit, playerDeath: Sprites para animar al jugador en pantalla.
6. projectileImg y projectileDestImg: Sprites para animar los proyectiles.

Audio

1. ost: Música de fondo para todas las pantallas menos la de muerte.
2. arrow: Sonido para la flecha.
3. hit: Sonido para golpes al jugador.
4. death: Música que se reproduce al llegar a la pantalla de muerte.

Variables para el apartado gráfico

Variables para botones, tamaño de la ventana, rectángulos, colores, etc.

1. width, height: Dimensiones del monitor, en píxeles.
2. screen: Superficie para la ventana base del programa.

Colores y superficies para los valores alpha

Valores dados en RGBA.

1. white = (255, 255, 255)
2. black = (0, 0, 0)
3. red = (255, 0, 0)
4. blue = (84, 137, 206)
5. yellow = (244, 252, 15)
6. fadedWhite = (255, 255, 255, 100)
7. fadedRed = (142, 0, 0, 100)
8. alpha_surf: Superficie para aplicar una capa semitransparente a imágenes.

Colores para los botones

1. buttonBg = (206, 163, 84)
2. buttonBorder = (206, 200, 84)

3. selectedButtonBg = (84, 137, 206)
4. selectedButtonBorder = (84, 184, 206)
5. audioOff = (252, 55, 68)
6. audioOn = (41, 173, 50)

Variaciones de sprites para el proyectil

Se necesitan 8 variaciones del sprite del proyectil (1 para cada dirección), que se rotarán usando *pygame.transform.rotate(superficie, ángulo)*:

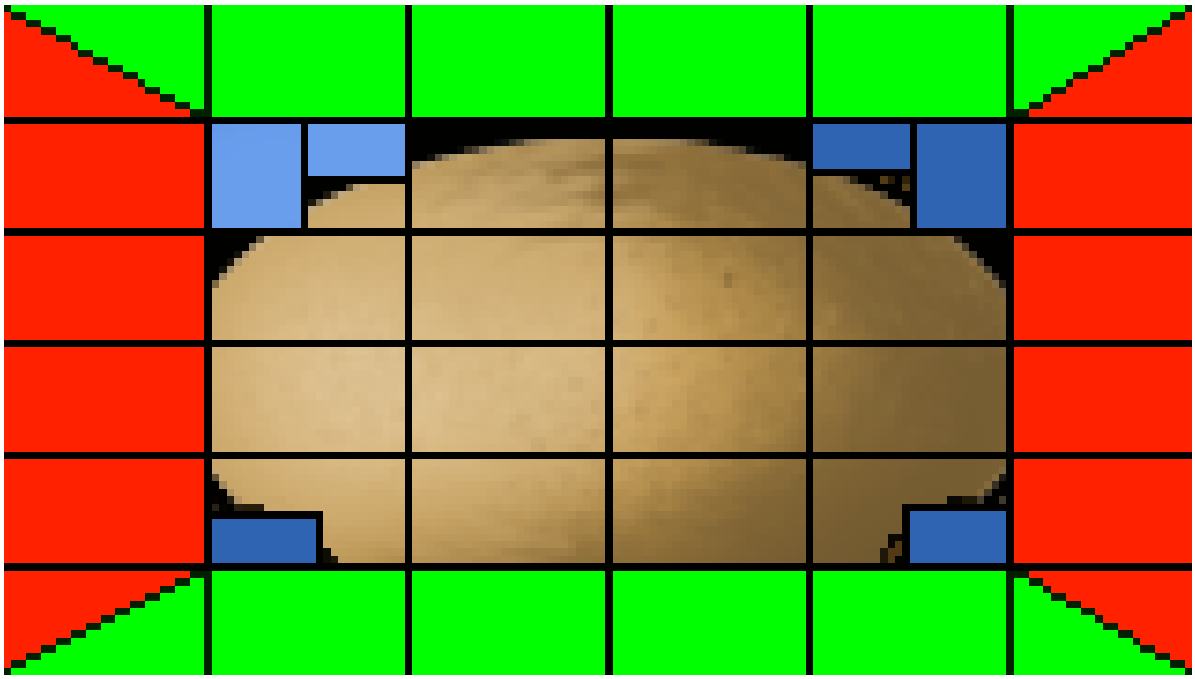
1. projSprite0 = projectileImg
2. projSprite45 = pygame.transform.rotate(projSprite0, 45)
3. projSprite90 = pygame.transform.rotate(projSprite0, 90)
4. projSprite135 = pygame.transform.rotate(projSprite0, 135)
5. projSprite180 = pygame.transform.rotate(projSprite0, 180)
6. projSprite225 = pygame.transform.rotate(projSprite0, 225)
7. projSprite270 = pygame.transform.rotate(projSprite0, 270)
8. projSprite315 = pygame.transform.rotate(projSprite0, 315)
9. projDestSprite0 = projectileDestImg
10. projDestSprite45 = pygame.transform.rotate(projDestSprite0, 45)
11. projDestSprite90 = pygame.transform.rotate(projDestSprite0, 90)
12. projDestSprite135 = pygame.transform.rotate(projDestSprite0, 135)
13. projDestSprite180 = pygame.transform.rotate(projDestSprite0, 180)
14. projDestSprite225 = pygame.transform.rotate(projDestSprite0, 225)
15. projDestSprite270 = pygame.transform.rotate(projDestSprite0, 270)
16. projDestSprite315 = pygame.transform.rotate(projDestSprite0, 315)

Variables para los rectángulos de los botones, vida y energía

- playRect, diffRect, audioRect, exitRect: Son todos objetos tipo *Rect*. Rectángulos para los botones.
- hpBar, spBar: Objetos de tipo *Rect* para las barras de la vida y de la energía.

Variables para los rectángulos que delimitan el área no jugable

El área no jugable se divide en 10 rectángulos:



Rectángulos que delimitan el área no jugable.

1. rect1…rect10: Rectángulos individuales para delimitar el movimiento del jugador y de los proyectiles.
2. border: Lista que almacena los diez rectángulos.

Variables para las fuentes de texto

1. titleFont: Objeto *pygame.Font* para el texto usado en los títulos.
2. titleFontSurface, pauseFontSurface, deathFontSurface: Superficies para el título de cada respectiva pantalla.
3. textFont: Fuente para el resto del texto usado en el juego (botones, barras de vida y energía, puntuación, etc).

Funciones

En Config.py hay varias funciones para uso general:

Funciones para el control de la dificultad

1. decreaseDif: Decrementa el valor de *gameDifficulty*. Si ya se encuentra en 1 (Fácil), aumenta en 1 el valor de *babyModeCounter*. Si *babyModeCounter* es 5, *gameDifficulty* se asigna a 0. En cualquier caso, al final se llama a *assignDiffName()*.
2. increaseDif: Incrementa el valor de *gameDifficulty*. Si ya se encuentra en 3 (Difícil), aumenta en 1 el valor de *bulletHellCounter*. Si *bulletHellCounter* es 5, *gameDifficulty* se asigna a 4. En cualquier caso, al final se llama a *assignDiffName()*.
3. assignDiffName: Cambia el valor de *gameDifficultyName* dependiendo de la dificultad seleccionada actualmente, indicada por *gameDifficulty*.

Funciones para el control de los botones y dibujar en pantalla

1. drawButton: Dibuja un botón en pantalla, dado: el tamaño, la superficie en la que dibujarlo, el texto a escribir en él, el color del texto, la posición donde colocarlo, el color, la anchura (si es 0 es un rectángulo lleno, si es > 0 indica cómo de ancho es el borde), el radio del borde, el color del borde, si está activo o no, y el color a usar si está activo.
2. drawButtons: Dibuja todos los botones necesarios en pantalla, dado: el texto del primer botón, si existe un botón para las dificultades, el texto del segundo botón, el texto del tercer botón, el texto del cuarto botón, si se encuentra en la pantalla de muerte, y si se encuentra en la pantalla de créditos. Dependiendo de la situación tendrá que dibujar tres, cuatro o cinco botones con textos variados.
3. drawSelectedButton: Dado una selección, activará el botón seleccionado.
4. drawUnselectedButton: Dado una selección, desactivará el botón que ya no estará seleccionado.
5. drawControls: Dibuja en pantalla un recuadro con los controles útiles para el usuario.

Funciones auxiliares

1. generateProjectile: Genera un proyectil de forma aleatoria con la velocidad indicada.
2. checkBounds: Comprueba que un rectángulo dado no colisione con ninguno de los rectángulos que delimitan el área jugable.
3. cleanConsole: Limpia (salta 20 líneas) la consola.

Objetos

Player

Atributos

1. playerHitBox: Rectángulo para la hitbox del jugador. Dimensiones: 1/20 de la anchura del monitor, tanto de altura como de anchura.
2. health: Entero para la vida. Por defecto en 100.
3. stamina: Entero para la energía. Por defecto en 100.
4. speed: Tupla de enteros para la velocidad del jugador. Por defecto (0, 0).
5. alive: Bool para controlar el estado del jugador (vivo/muerto).

Métodos

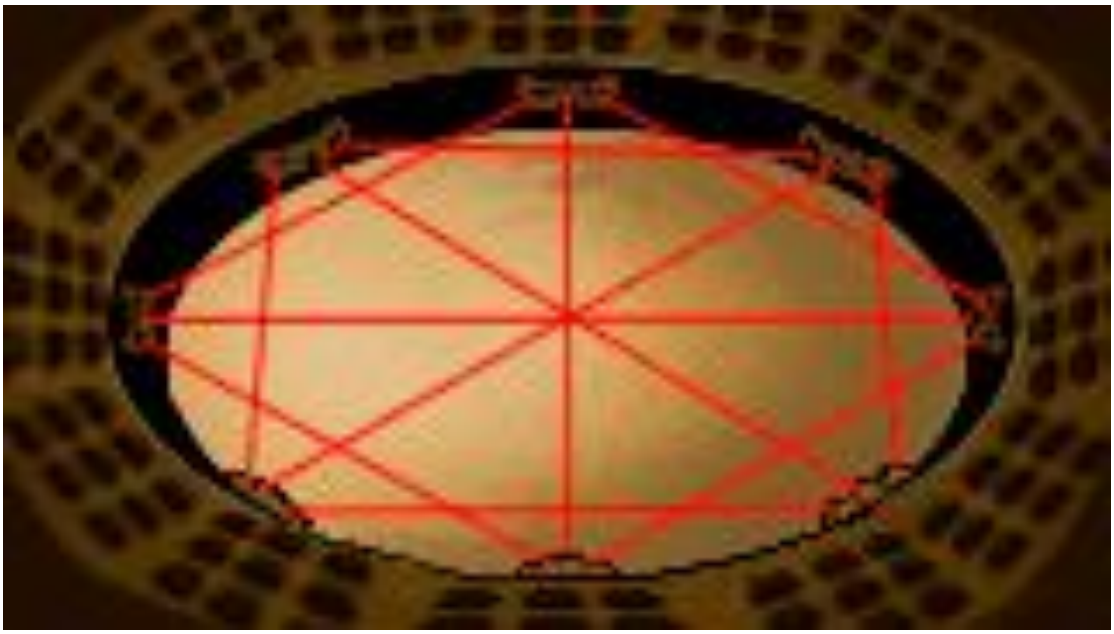
1. isAlive: Devuelve True si está vivo, False en caso contrario.
2. die: Establece alive a False.
3. updateHP: Suma a la vida del jugador la cantidad indicada (siempre se le dan números negativos).
4. updateSP: Suma a la energía del jugador la cantidad indicada.
5. stop: Establece la velocidad del jugador a (0, 0).
6. isStopped: Devuelve True si speed es (0, 0), False en caso contrario.
7. updatePos: Mueve el rectángulo que delimita la hitbox del jugador en la dirección indicada por speed.
8. animate: Dibuja en el rectángulo de la hitbox el sprite del jugador correspondiente.
9. animateHit: Dibuja en el rectángulo de la hitbox el sprite para el golpe del jugador. También dibuja la cantidad de daño recibida al lado del jugador.

10. drawHPBar: Dibuja en pantalla la barra de HP del jugador.
11. drawSPBar: Dibuja en pantalla la barra de SP del jugador.
12. getRect: Devuelte la hitbox del jugador.

Projectile

Atributos

1. projectileImg: Sprite correspondiente del proyectil.
2. projectileDestImg: Sprite correspondiente para la destrucción del proyectil.
3. projectileHitBox: Rectángulo para la hitbox del proyectil. La altura será de 1/40 de la anchura del monitor, y la anchura de 1/20 de la anchura del monitor.
4. speed: Tupla de enteros para la velocidad del proyectil. Inicialmente se le asigna una velocidad basada en la dada al constructor.
5. gate: Puerta en la que aparecerá el proyectil. Es un entero generado de forma pseudoaleatoria, entre 1 y 8 (ambos inclusive). Hay un total de 8 puertas, colocadas cada 45º.
6. dir: Dirección en la que se moverá el proyectil. Es un entero generado de forma pseudoaleatoria, entre 1 y 3 (ambos inclusive). Hay 3 direcciones para cada puerta, siendo la 2 la línea que une el centro de la arena con la puerta; la dirección 1 será una 45º superior a esta, y la 3 estará a -45º de la 2.



Direcciones de los proyectiles.

Métodos

1. assignSpeed: Dada una velocidad, usa gate y dir para crear un vector de velocidad en el que se moverá el proyectil.
2. updatePos: Actualiza la posición del proyectil, moviendo la hitbox en la dirección indicada por la velocidad.
3. checkCollision: Dado el rectángulo del jugador, comprueba si se ha producido una colisión. Devuelve True si se ha producido, False si no.

4. assignInitialPos: Asigna la posición inicial al proyectil dependiendo de la gate y dir asignadas.
5. animate: Dibuja en la hitbox del proyectil el sprite en la orientación correspondiente.
6. destroy: Dibuja en la hitbox del proyectil el sprite de destrucción de la orientación correspondiente, y elimina la referencia a la hitbox del proyectil.

Archivo Main.py

Sólo contiene una función: start(). Esta función llama a Game.run().

Archivo Game.py

Este archivo tiene dos funciones clave:

Game.run()

Es el bucle principal del juego. En este bucle se comprueba si se sigue ejecutando el programa, se vuelve a jugar, o se vuelve al menú principal.

Usando las variables *playAgain* y *playState* se controla si se ha seleccionado volver a jugar al morir, si se ha elegido volver al menú principal en algún momento, o si se ejecuta el juego (*Game.start()*).

Game.start(dif)

Comienza el juego con una dificultad indicada. En esta función se asigna la velocidad de proyectil, la velocidad del jugador, y el daño por colisión usando funciones auxiliares. En esta función se controla el movimiento del jugador y todos los posibles eventos que puedan ocurrir durante la ejecución del bucle. Aquí es donde se suman los puntos.

Funciones auxiliares

1. assignProjectileSpeed: Asigna la velocidad de los proyectiles.
2. assignPlayerSpeed: Asigna la velocidad del jugador.
3. assignHitHP: Asigna la vida a perder por la colisión entre proyectil y jugador.
4. drawPoints: Dibuja en pantalla la puntuación actual del jugador.

Archivo MainTitle.py

MainTitle.run()

En esta función se encuentra el bucle del menú principal. En ella se encuentra la lógica para el control de los botones. Si se selecciona “Jugar”, devuelve 1; si se selecciona “Salir”, devuelve 0.

Funciones auxiliares

1. redraw: Función para redibujar los componentes de la pantalla (botones y texto).

Archivo CreditScreen.py

CreditScreen.run()

Dibuja en pantalla los créditos. Si se presiona X, se vuelve a MainTitle.run(), devolviendo 1.

Archivo PauseScreen.py

PauseScreen.run()

Función para el bucle en el que se controla la pantalla de pausa. Si se presiona “Continuar” o ESC, se vuelve a Game.start(), devolviendo 2; si se elige “Salir”, devuelve 0; y si se elige “Menú principal”, devuelve 1.

Archivo DeathScreen.py

DeathScreen.run()

Función para el bucle en el que se controla la pantalla de fin del juego. Desde ella se puede elegir volver a jugar, volver al menú principal, o salir. La función devolverá respectivamente 2, 1, y 0, dependiendo de la elección. En ella se muestra la puntuación obtenida y la dificultad en la que se ha jugado.