

FNAF

Five Nights at Freddy's

DAM - 0487. Entorns de desenvolupament

Denise Name | Javier Tituaña

Índice

1. Descripción del Juego	3
2. Cómo Jugar	3
3. Estructura del Proyecto y Clases	4
4. Stats y Recursos	4
5. Condiciones de Victoria y Derrota	5
6. Progresión de las Noches	5
7. Guía de los Animatronics	6
8. Análisis del Código	7
8.1 Main.java	7
8.2 Noches.java	8
8.3 Animatronics.java	9
9. Sistema de Aleatoriedad y Probabilidades	9
10. Consejos y Estrategia	9

REPOSITORIO GITHUB: <https://github.com/javiitc/FNAF.git>

CANVA PRESENTACIÓN: <https://www.canva.com/design/DAHAQqjyIOA>

1. Descripción del Juego

FNAF es inspirado en Five Nights at Freddy's. El jugador asume el papel de un guardia de seguridad nocturno en la pizzería de Freddy Fazbear. A lo largo de cinco noches, el jugador debe sobrevivir los encuentros con los personajes animatronics tomando decisiones rápidas mientras administra recursos limitados.

El juego es completamente basado en la consola: toda la interacción se realiza mediante texto y entrada numérica. No hay gráficos, solo texto mostrado en la terminal. Cada noche introduce nuevos animatronics y aumenta la dificultad, culminando con la aparición de Golden Freddy en la Noche 5.

2. Cómo Jugar

Ciclo de Juego

1. El juego comienza pidiendo el nombre del jugador.
2. El jugador es asignado como guardia nocturno y comienza la Noche 1.
3. Cada noche ocurren una serie de eventos aleatorios: los animatronics se acercan o se activan eventos bonus.
4. En cada encuentro, el jugador selecciona una acción (esconderse, cerrar la puerta o no hacer nada).
5. Una vez que se han usado todos los movimientos de una noche, si el jugador sigue vivo, llegan las 6 AM y comienza la siguiente noche.
6. Si el jugador sobrevive las cinco noches, gana el juego.

3. Estructura del Proyecto y Clases

El proyecto está compuesto por tres clases de Java:

Clase	Rol	Descripción
Main.java	Punto de Entrada	Contiene el método main(). Gestiona la entrada del nombre del jugador y controla el bucle principal del juego a lo largo de las cinco noches.
Noches.java	Gestor de Noches	Define cada una de las cinco noches. Gestiona la cordura, la energía, los límites de movimientos y la selección aleatoria de eventos por noche.

Animatronics.java	Gestor de Eventos	Contiene todos los métodos de encuentro con los animatronics. Procesa las opciones del jugador, calcula los resultados usando aleatoriedad y actualiza el estado del juego.
--------------------------	-------------------	---

4. Stats y Recursos

El jugador tiene dos recursos clave que se controlan a lo largo de todo el juego (en todas las noches). Ambos comienzan en 150.

Stat	Descripción
Cordura (cordura)	Representa el estado mental del jugador. Disminuye en 5 después de cada evento durante una noche. Ver la televisión restaura +15.
Energía (energia)	Representa la energía eléctrica de la oficina. Cerrar una puerta o cortina consume 10 de energía. Freddy drena automáticamente el 15% de la energía actual cuando aparece. Ver la televisión también consume 10 de energía. Si la energía llega a 0, el jugador muere inmediatamente.

Ambos stats persisten a lo largo de las cinco noches. La energía y la cordura consumidas en la Noche 1 se mantienen en la Noche 2 y más adelante. Los jugadores deben ser cuidadosos con los recursos desde el principio.

5. Condiciones de Victoria y Derrota

Condición	Cómo Ocurre
VICTORIA	El jugador sobrevive las cinco noches (todos los movimientos de cada noche se usan sin morir).
GAME OVER	El jugador muere durante cualquier noche. La muerte puede ocurrir por:

- Un animatronic mata al jugador después de un fallo en esconderse o no hacer nada (determinado por aleatoriedad).

- La energía llega a 0: la oficina queda a oscuras y los animatronics atacan.
- Aparece Golden Freddy y randomizer de muerte es desfavorable (75% de probabilidad en la Noche 5).
- El jugador ingresa una entrada no válida durante un encuentro con un animatronic (se trata como un pánico, lo que resulta en la muerte).

6. Progresión de las Noches

Cada noche aumenta en dificultad. El número de movimientos (eventos) crece y se introducen nuevos animatronics. La tabla a continuación muestra exactamente qué cambia cada noche:

Noche	Movimientos	Eventos Posibles	Nuevo Elemento
1	3	Bonnie	Bonnie (único enemigo)
2	4	Bonnie, Chica, TV	Chica + evento TV
3	5	Bonnie, Chica, Foxy, TV	Foxy
4	5	Bonnie, Chica, Foxy, Freddy, TV	Freddy
5	6	Bonnie, Chica, Foxy, Freddy, Golden Freddy, TV	Golden Freddy

7. Guía de los Animatronics

Cada animatronic se comporta de manera diferente. La tabla a continuación muestra el resultado de cada acción del jugador por animatronic. Los porcentajes representan la probabilidad de muerte para cada opción.

Animatronic	Esconderse	Cerrar Puerta / Cortina	No Hacer Nada
Bonnie	25% muerte	Seguro (-10 energía)	25% muerte
Chica	25% muerte	Seguro (-10 energía)	25% muerte
Foxy	50% muerte	Seguro (-10 energía)	~33% muerte
Freddy	– (evento auto)	– (evento auto)	– (evento auto)
Golden Freddy	– (sin opción)	– (sin opción)	75% muerte (auto)

Detalles Individuales de Cada Animatronic

Bonnie

Bonnie es el primer animatronic al que se enfrenta el jugador (Noche 1). Se acerca a la oficina y el jugador debe reaccionar. Esconderse detrás de la máscara de Freddy o no hacer nada implica un 25% de probabilidad de muerte. Cerrar la puerta es la única opción garantizada de seguridad, pero consume 10 de energía.

Chica

Chica se comporta de manera idéntica a Bonnie en términos de mecánicas y probabilidades. Se introduce en la Noche 2. Las mismas tres opciones están disponibles con los mismos niveles de riesgo.

Foxy

Foxy es notablemente más peligroso que Bonnie o Chica. Corre por el pasillo a alta velocidad. Esconderse tiene un 50% de probabilidad de muerte (el doble de riesgo), y no hacer nada también conlleva un riesgo más alto (~33%). Cerrar la cortina sigue siendo la única opción segura. Foxy se introduce en la Noche 3.

Freddy

Freddy es una amenaza pasiva: no puede ser engañado por la máscara de Freddy y no ofrece opciones al jugador. Cuando aparece, quita automáticamente el 15% de la energía actual del jugador. Esto significa que cuanto más energía tenga el jugador, más le toma Freddy. Se introduce en la Noche 4.

Golden Freddy

Golden Freddy es el animatronic más lethal del juego. Cuando aparece, el jugador queda paralizado y no puede actuar. Una tirada aleatoria determina la supervivencia: hay un 75% de

probabilidad de muerte y solo un 25% de que el encuentro haya sido una alucinación. Aparece exclusivamente en la Noche 5.

Evento TV (Bonus)

El evento de la TV no es un animatronic sino una oportunidad de bonus. El jugador encuentra un control remoto y puede elegir ver la televisión. Ver la TV consume 10 de energía pero restaura 15 de cordura, siendo una forma útil de mantener los niveles de cordura saludables. El jugador también puede rechazarlo.

8. Análisis del Código

8.1 Main.java – Punto de Entrada del Juego

Declaración de Variables

Scanner sc	→ Lee la entrada del usuario desde la consola.
Random random	→ (declarado pero no usado aquí; se usa en Noches).
Noches noches	→ Crea el objeto Noches que gestiona las cinco noches.
int contador	→ Controla el número de noche actual (del 1 al 5).
boolean juego	→ Controla el bucle principal (true = el juego sigue en curso).
String nombre	→ Almacena el nombre del jugador.

Bloque de Entrada del Nombre

El juego le pide al jugador que ingrese su nombre. Si la entrada está vacía o se lanza una excepción, el nombre por defecto es "William Afton", una referencia a la lore de FNAF (creo que era Michael no????)

Bucle Principal del Juego

Un bucle while se ejecuta mientras juego sea true. Dentro del bucle, una cadena if-else verifica el valor de contador para determinar qué método de noche llamar (noche1() hasta noche5()). Si contador supera 5, el jugador gana. Si algún método de noche retorna false (el jugador murió), el juego termina con un mensaje de GAME OVER.

8.2 Noches.java – Gestor de Noches

Noches.java contiene la lógica de cada una de las cinco noches. Almacena los stats compartidos (cordura y energía) y define el bucle de eventos para cada noche.

Variables Compartidas

int cordura = 150	→ Cordura del jugador. Compartida entre todas las noches.
int energia = 150	→ Energía del jugador. Compartida entre todas las noches.

Estructura del Bucle de Noche (aplica a todas las noches)

Cada método de noche sigue el mismo patrón general:

1. Imprimir el número de noche y el mensaje del Phone Guy.
2. Inicializar conVida (vivo = true) y movesLeft (varía por noche).

3. Entrar en un bucle while que se ejecuta mientras conVida sea true Y movesLeft > 0.
4. Mostrar los stats actuales de Cordura y Energía.
5. Generar un evento aleatorio usando random.nextInt(n), donde n es el número de eventos posibles para esa noche.
6. Llamar al método correspondiente en Animatronics.java según el valor aleatorio.
7. Si el método del evento retorna false, el jugador está muerto: salir del bucle.
8. De lo contrario, disminuir cordura en 5, decrementar movesLeft y verificar si energia ha llegado a 0.
9. Retornar conVida a Main.java para indicar si el jugador sobrevivió.

Diferencias Específicas por Noche

- Noche 1: Solo eventos de Bonnie. movesLeft = 3. Sin aleatoriedad: Bonnie siempre aparece.
- Noche 2: Elección aleatoria entre Bonnie, Chica o TV. movesLeft = 4.
- Noche 3: Agrega a Foxy al grupo aleatorio. movesLeft = 5.
- Noche 4: Agrega a Freddy. movesLeft = 5.
- Noche 5: Agrega a Golden Freddy. movesLeft = 6.

8.3 Animatronics.java – Gestor de Eventos

Esta clase contiene un método separado para cada encuentro con un animatronic y el evento de la TV. Cada método gestiona la visualización del mensaje, la lectura de la opción del jugador, las tiradas de resultado y la actualización del estado del juego.

Variables Compartidas

Scanner sc	→ Lee la entrada del jugador.
Random random	→ Se usa para todas las tiradas de muerte/supervivencia.
int choice	→ Almacena la opción seleccionada por el jugador (1, 2 o 3).
boolean isAlive	→ Controla si el jugador sigue vivo después del evento.

bonnieEvent() / chicaEvent()

Ambos métodos siguen la misma estructura y probabilidades. El flujo es:

1. Mostrar el mensaje del encuentro y las tres opciones.
2. Leer la entrada. Si es no válida, retornar false (el jugador muere de pánico).
3. Procesar la opción mediante un switch:
 - Caso 1 (Esconderse): Se tira el arreglo chanceDeath [1, 0, 0, 0] con nextInt(3), es decir, índices 0, 1 o 2. Índice 0 = muerte (1), índices 1 o 2 = supervivencia (0). Esto da una probabilidad efectiva de muerte de 1 en 3 de los elementos accesibles.
 - Caso 2 (Cerrar Puerta): Se resta 10 a energía. Siempre seguro.
 - Caso 3 (No Hacer Nada): Misma tirada que Esconderse, pero con lógica invertida: obtener 0 significa muerte.

foxyEvent()

Foxy usa dos arreglos separados para mayor peligro:

```
chanceDeath = {1, 0, 1, 0} → Se usa para Esconderse (Caso 1).
chanceLife  = {1, 0, 1, 0, 1, 0} → Se usa para No Hacer Nada (Caso 3).
```

Para esconderse, la tirada usa `nextInt(3)` en `chanceDeath`: dos de los tres índices accesibles contienen el valor 1 (muerte), haciendo la muerte más probable. Para no hacer nada, la tirada usa `nextInt(n)` en un arreglo de 6 elementos.

freddyEvent()

Freddy no ofrece interacción al jugador. Cuando se activa, el método calcula inmediatamente el 15% de la energía actual y lo resta. Si la energía quedaría por debajo de cero, se limita a 0. El método retorna void (no es posible morir solo por Freddy).

goldenFreddyEvent()

Golden Freddy tampoco ofrece opciones al jugador. El jugador queda "paralizado" y una sola tirada determina su destino usando `chanceDeath = {1, 1, 0, 1}` con `nextInt(3)`. Dos de los tres índices accesibles son 1 (muerte), dando aproximadamente un 67% de probabilidad de muerte desde los elementos accesibles (aunque el arreglo sugiere una intención del 75% en total).

tvEvent()

El evento de la TV le da al jugador la opción de ver la televisión o no. Si el jugador elige verla, energía se reduce en 10 y cordura se aumenta en 15. Este método retorna void ya que no puede matar al jugador.

9. Sistema de Aleatoriedad y Probabilidades

El juego usa la clase `java.util.Random` de Java y arreglos de enteros predefinidos para simular resultados de azar:

```
int[] chanceDeath = {1, 0, 0, 0};
int deathRoll = chanceDeath[random.nextInt(3)];
// nextInt(3) retorna 0, 1 o 2
// Valores del arreglo en esos índices: 1, 0, 0
// Resultado: 1 = muerte, 0 = supervivencia
```

El valor 1 en el arreglo representa muerte y 0 representa supervivencia. El rango pasado a `nextInt()` determina qué índices son accesibles, lo cual a su vez controla la probabilidad efectiva.

10. Consejos y Estrategia

- Cerrar las puertas es la opción más segura contra Bonnie, Chica y Foxy, pero la energía es limitada. Úsala con cuidado.

- Freddy quita automáticamente el 15% de la energía actual. Cuanto más energía tengas, más te toma. Nada puede detenerlo.
- Ve la televisión cuando sea posible. Consume 10 de energía pero restaura 15 de cordura, ayudándote a resistir durante noches más largas.
- Golden Freddy es casi inevitable. Si aparece, hay aproximadamente un 67-75% de probabilidad de muerte. La supervivencia depende de la suerte.
- La energía es el recurso más crítico. Quedarte sin energía es una derrota inmediata, por lo que prioriza conservarla en las noches posteriores.
- Una entrada no válida (no numérica) cuenta como muerte durante un encuentro con un animatronic. Siempre ingresa un número válido (1, 2 o 3).