Operación Valquiria

Diseño y Análisis del Videojuego

Proyecto 2025-2

Brayan Silva | Hamilton Suárez | Informatica 2 | Universidad de Antioquia

1. INTRODUCCIÓN

Este documento desarrolla el Momento II del proyecto final “Operación Valquiria”, un videojuego educativo basado en los acontecimientos del 20 de julio de 1944, cuando el coronel Claus von Stauffenberg lideró el intento de asesinato contra Adolf Hitler.

Objetivo del proyecto

Desarrollar un videojuego funcional en C++ utilizando Qt que ponga en práctica los conocimientos de:

- Programación Orientada a Objetos (POO)
- Memoria dinámica
- Interfaz gráfica de usuario (GUI)
- Implementación de agentes inteligentes

Alcance del Momento II

Este documento incluye:

- Descripción técnica de los tres niveles del videojuego
- Explicación de las mecánicas principales y físicas aplicadas
- Diseño de los agentes inteligentes que interactúan con el jugador
- Bocetos o vistas esquemáticas de cada nivel
- Plan de sprites (gráficos) y uso de contenedores STL

El juego está dividido en tres fases, representando distintos momentos del complot histórico:

1. Nivel 1: Infiltración al cuartel
2. Nivel 2: Ensamblaje del dispositivo explosivo
3. Nivel 3: Escape aéreo del campo enemigo

2. DESCRIPCIÓN DETALLADA DE LOS NIVELES

2.1 Nivel 1: Infiltración

Contexto narrativo:
El coronel Stauffenberg debe ingresar al cuartel general conocido como *La Guarida del Lobo* para obtener los planos del edificio sin ser descubierto por los guardias. Este nivel combina sigilo, observación y desplazamiento básico.

Tipo de vista:
2D lateral con scroll horizontal limitado. La cámara sigue al jugador dentro del cuartel, mostrando habitaciones y pasillos conectados.

Escenario principal:

- Pasillos de concreto con luces tenues
- Habitaciones conectadas por puertas
- Cajas y muebles para ocultarse
- Guardias patrullando con linternas
- Puntos de control (checkpoints)

—----------

Jugabilidad básica

Acciones del jugador:

- **Moverse:** izquierda/derecha
- **Agacharse:** tecla `C` (reduce visibilidad)
- **Correr:** tecla `Shift` (más rápido, pero genera ruido)
- **Interactuar:** tecla `E` (abrir puertas o recoger objetos)
- **Ocultarse:** acercarse a cajas o sombras (temporalmente invisible)

Objetivo del nivel:
Llegar hasta la oficina central, recoger los planos y salir sin ser detectado.

Enemigos (guardias)

Los guardias son agentes inteligentes básicos con comportamiento autónomo.

- Patrullan entre puntos predefinidos (A → B → C → A).
- Poseen un campo de visión limitado (frente a ellos).
- Si ven o escuchan al jugador, lo persiguen por un corto tiempo.
- Si lo pierden de vista, regresan a su ruta normal.

Estados simples del guardia:

- Patrullando: se mueve entre puntos.
- Alertado: vio movimiento o escuchó ruido.
- Persiguiendo: corre hacia el jugador.
- Volviendo: retoma su ruta original.

Este comportamiento se controla mediante una máquina de estados sencilla (FSM).

Objetos interactivos

- Puertas: pueden estar abiertas, cerradas o bloqueadas.
- Llaves: abren puertas específicas.
- Zonas de sombra: reducen la probabilidad de detección.
- Puntos de control: guardan progreso al pasar por ciertas zonas.

Físicas utilizadas

1. **Movimiento rectilíneo uniforme**
   Ecuación: $x(t) = x_0 + v \cdot t$
   - Modela el desplazamiento del jugador y de los guardias.
   - Velocidad promedio: jugador = 80 px/s, guardia = 60 px/s.

2. **Campo visual (detección angular)**
   Ecuación: $\theta(t) = \theta_0 + \omega \cdot t$
   - Permite rotar el cono de visión del guardia lentamente.
   - Simula el efecto de “mirar hacia los lados”.

3. **Cálculo de distancia**
   Ecuación: $d = \sqrt{[(x_1 - x_2)^2 + (y_1 - y_2)^2]}$
   - Determina si el jugador está dentro del rango de detección.
   - Si $d < 150$ px y el jugador está visible → detección.

Condiciones de juego

Victoria: Obtener los planos y regresar a la salida sin ser capturado.

Derrota: Ser detectado 3 veces o alcanzado por un guardia.

Puntuación:

- +1000 puntos si completa sin detecciones
- +500 puntos por sigilo perfecto (sin correr ni alertar guardias)

2.2 Nivel 2: Ensamblaje y preparación de la bomba

Contexto narrativo:
En una sala aislada del cuartel, el coronel **Stauffenberg** debe **ensamblar la bomba** dentro de un maletín antes de asistir a la reunión con Hitler.
El proceso requiere precisión y rapidez: cada componente debe ser colocado en el orden correcto, y cualquier error podría arruinar toda la operación.
Mientras el reloj avanza, la tensión aumenta y cada segundo cuenta.

Tipo de vista:

Cenital fija (desde arriba del escritorio).
El entorno muestra una mesa con las piezas dispersas y un maletín abierto al centro. No hay movimiento de cámara; el jugador interactúa únicamente con los objetos sobre la mesa.

Escenario principal:

- Fondo: escritorio metálico con herramientas.
- Objetos interactivos: detonador, carga explosiva, temporizador, maletín.
- Reloj/temporizador visible en pantalla.
- Luz tenue, ambiente silencioso con sonidos mecánicos al interactuar.

—----------

Jugabilidad básica

Acciones del jugador:

- **Arrastrar y soltar** cada pieza al área correspondiente (drag & drop).
- **Orden lógico de ensamblaje:**
    1. Colocar el detonador.
    2. Insertar la carga explosiva.
    3. Ajustar el temporizador.
    4. Cerrar el maletín.

- **Penalizaciones:**
    1. Pieza mal colocada: -5 segundos
    2. Orden incorrecto: -5 segundos
    3. Colocación correcta: +2 segundos (bonus)

Objetivo del nivel:
Completar el ensamblaje del artefacto explosivo antes de que el tiempo llegue a cero, asegurando que cada pieza esté correctamente posicionada.

Mecánica del temporizador

El jugador dispone de **90 segundos** para finalizar la tarea.
El cronómetro decrece en tiempo real y aparece visible en la esquina superior de la pantalla.

Cada error resta 5 segundos; las colocaciones correctas pueden otorgar pequeñas bonificaciones de tiempo (por ejemplo, +2 segundos).

Físicas implementadas

1. **Movimiento lineal (desplazamiento de piezas)**
   Ecuaciones:
   $x(t) = x_0 + v_0 t \qquad y(t) = y_0 + v_0 t$
   Modela el movimiento de las piezas cuando son arrastradas con el cursor o con las teclas direccionales.
   Permite un desplazamiento suave desde la posición inicial hasta la zona de encaje.

2. **Colisión circular (encaje de piezas)**
   Ecuación:
   $(x_1 - x_2)^2 + (y_1 - y_2)^2 < (r_1 + r_2)^2$
   Determina si una pieza fue colocada correctamente dentro del área de encaje asignada.
   Si la condición se cumple, la pieza queda fija; de lo contrario, regresa a su posición original y se aplica una penalización.

3. **Temporizador decreciente (control de tiempo)**

   - Variable: `timer = 90 - elapsed_time`
   - Si `timer <= 0`, el jugador pierde el nivel.
   - En C++/Qt puede implementarse con `QTimer` y señales `timeout()` para actualizar el tiempo restante cada segundo.

Retos principales

- Encajar todas las piezas correctamente antes del límite de tiempo.
- Recordar el orden de ensamblaje exacto.
- Mantener precisión en el movimiento para no perder tiempo por errores de colocación.

Condiciones de juego

Victoria: Todas las piezas ensambladas correctamente y el maletín cerrado antes de que el temporizador llegue a cero.

Derrota: Tiempo agotado o tres errores consecutivos de colocación.

Indicadores visuales y sonoros:

- Sonido metálico al colocar una pieza correctamente.
- Sonido de chispa o “error” al colocar mal una pieza.
- Cambio de color en el temporizador (verde → amarillo → rojo) para indicar urgencia.

Funcionamiento general

El jugador observa los componentes del artefacto dispuestos sobre la mesa.
Utilizando el ratón o el teclado, debe arrastrar cada pieza al punto exacto del maletín.
Cada colocación correcta emite un sonido característico; los errores penalizan el tiempo restante.
Al completar el ensamblaje, se muestra una animación corta de Stauffenberg cerrando el maletín, que da paso directo al nivel del escape aéreo.

2.3 Nivel 3: Escape terrestre y aéreo

Contexto narrativo

Tras dejar el maletín en la sala de reuniones, el coronel Stauffenberg debe escapar del cuartel antes de que la explosión ocurra. Primero debe correr hacia la pista donde lo espera un avión. Una vez en el aire, es perseguido por misiles enemigos que intentan derribarlo.

Este nivel tiene DOS FASES:

FASE 1: Escape Terrestre (30-40 segundos)

Tipo de vista: Lateral con scroll horizontal

Escenario:

- Pasillos y exteriores del cuartel
- Guardias alertados corriendo
- Obstáculos en el suelo (escombros, vallas)
- Pista de aterrizaje al final

—----------

Jugabilidad:

- **Correr:** automático hacia la derecha
- **Saltar:** tecla `Espacio` (evitar obstáculos y guardias)
- **Agacharse:** tecla `C` (pasar bajo barreras)
- **Sprint:** tecla `Shift` (burst de velocidad)

Objetivo: Llegar al avión antes de que te alcancen o choque con obstáculos

Físicas Fase 1:

1. Movimiento parabólico (salto)

$y(t) = v_0 \cdot \sin(\theta) \cdot t - \frac{1}{2} \cdot g \cdot t^2$ $x(t) = v_0 \cdot \cos(\theta) \cdot t$

- Controla la trayectoria del salto del personaje
- v_0 = 200 px/s, θ = 60°, g = 980 px/s²
- Permite calcular altura máxima y distancia del salto

2. Movimiento constante (desplazamiento base)

$x(t) = v \cdot t$

- Desplazamiento horizontal automático
- v = 100 px/s (velocidad base de carrera)
- Con sprint: v = 150 px/s

FASE 2: Escape Aéreo (90-120 segundos)

Tipo de vista: Lateral con desplazamiento horizontal continuo (side scrolling)

El avión del jugador se mantiene centrado en pantalla, mientras el fondo y los enemigos se desplazan hacia la izquierda simulando el vuelo.

Escenario:

- Cielo con nubes en movimiento
- Torres y estructuras militares
- Globos aerostáticos
- Otros aviones enemigos
- Zona segura al final (frontera)

—----------

Jugabilidad:

- **Mover arriba/abajo:** ↑/↓ o `W/S`
- **Acelerar/frenar:** `A` y `D`
- **Evasión rápida:** `Espacio` (roll/giro)
- **Turbo:** `Shift` (velocidad temporal con recarga)

Objetivo: Evitar misiles y obstáculos hasta alcanzar la zona segura

Enemigos: Misiles Guiados (Agentes Inteligentes)

Los misiles son controlados por un agente autónomo con comportamiento de persecución predictiva.

Estados del misil:

1. **Búsqueda:** rastrea posición inicial del avión
2. **Persecución:** ajusta dirección en tiempo real
3. **Predicción:** estima posición futura del jugador
4. **Explosión:** se autodestruye si impacta o pierde al jugador

Lógica del agente:

cpp

```
predictedPos = player.pos + player.vel * leadFactor;

direction = normalize(predictedPos - missile.pos);

missile.pos += direction * missileSpeed * dt;
```

Aprendizaje básico:

- Si el misil falla varias veces, incrementa `leadFactor`
- Mejora su predicción con cada intento fallido

Físicas Fase 2

3. Movimiento sinusoidal (oscilación del avión)

$$y(t) = A \cdot \sin(\omega \cdot t + \varphi)$$

- Simula la oscilación natural del avión en vuelo
- A = 10 px (amplitud de oscilación)
- ω = 2π/3 rad/s (frecuencia)
- Crea sensación de vuelo realista

4. Trayectoria del misil (persecución direccional)

$$p_m(t + \Delta t) = p_m(t) + d \cdot v_m \cdot \Delta t$$

- d es el vector unitario hacia la posición predicha
- v_m = 180 px/s (velocidad del misil)
- Actualización cada frame del juego

5. Colisión circular (detección de impactos)

$$(x_m - x_p)^2 + (y_m - y_p)^2 < (r_m + r_p)^2$$

- Detecta colisiones entre avión y misiles
- r_m = 15 px (radio misil), r_p = 30 px (radio avión)

6. Desplazamiento del fondo (parallax scrolling)

$$x_{fondo} = x_{fondo} - v_{scroll} \cdot dt$$

- Genera sensación de movimiento hacia adelante
- v_scroll = 50 px/s (velocidad del scroll)
- Múltiples capas con velocidades diferentes (efecto parallax)

Retos principales

- **Fase 1:** Timing preciso de saltos, evitar guardias y obstáculos
- **Fase 2:** Esquivar misiles guiados, usar turbo estratégicamente, evitar colisiones

Condiciones de juego

Victoria:

- Completar Fase 1: llegar al avión
- Completar Fase 2: alcanzar zona segura sin impactos

Derrota:

- Fase 1: ser alcanzado por guardia o chocar 3 veces
- Fase 2: ser alcanzado por misil u obstáculo

Puntuación:

- Fase 1: +500 puntos por completar
- Fase 2: +100 puntos por cada misil esquivado
- Fase 2: +500 puntos si completa sin daños
- Fase 2: +200 puntos por tiempo récord (< 2:00 min)

3. Componente investigativo — Agentes inteligentes

El componente investigativo del videojuego *Operación Valquiria* se enfoca en la implementación de agentes autónomos sencillos, que dotan al entorno de comportamientos dinámicos y adaptativos.
Estos agentes son entidades no controladas por el jugador, capaces de percibir su entorno, tomar decisiones y actuar de manera independiente.

En el juego se desarrollan dos agentes inteligentes:

1. **Guardia patrullero** (Nivel 1 – Infiltración)

2. **Misil guiado** (Nivel 3 – Escape aéreo)

Ambos cumplen funciones distintas dentro de la narrativa, pero comparten una estructura basada en el ciclo de percepción–razonamiento–acción.

3.1 Guardia Patrullero — Nivel 1

Descripción general

El guardia es un agente reactivo que vigila áreas específicas del cuartel. Cuenta con un campo visual (ángulo y distancia) y capacidad de detectar sonidos.

Componentes del agente

A) PERCEPCIÓN

- **Campo visual:** cono de 90° con alcance de 150 px
- **Detección auditiva:** radio de 100 px para ruidos fuertes (correr)
- **Sensores:** posición del jugador, nivel de ruido, estado de puertas

B) RAZONAMIENTO Máquina de estados finitos (FSM) con 4 estados:

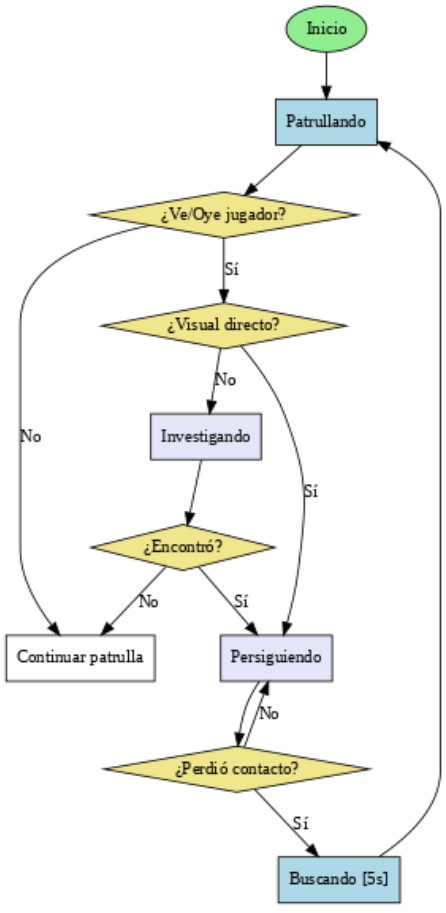
1. **PATRULLANDO**
    - Condición: estado por defecto
    - Decisión: moverse al siguiente punto de ruta
2. **INVESTIGANDO**
    - Condición: escuchó un ruido
    - Decisión: ir hacia la fuente del sonido
3. **PERSIGUIENDO**
    - Condición: vio al jugador directamente
    - Decisión: correr hacia última posición conocida
4. **BUSCANDO**
    - Condición: perdió de vista al jugador
    - Decisión: examinar árealocal por 5 segundos

C) ACCIÓN

- Ajustar velocidad según estado (patrulla=60px/s, persecución=100px/s)
- Rotar campo visual mientras patrulla
- Emitir alerta sonora al detectar
- Actualizar ruta de patrullaje

D) APRENDIZAJE

- Registra zonas donde detectó al jugador
- Aumenta tiempo de vigilancia en zonas sospechosas
- Reduce velocidad de patrullaje en áreas problemáticas



3.2 Misil Guiado — Nivel 3 (Fase 2)

Descripción general

El misil guiado es un agente predictivo que calcula la trayectoria de interceptación del avión. Anticipa la posición futura del jugador para maximizar la probabilidad de impacto.

Componentes del agente

A) PERCEPCIÓN

- **Posición del avión:** coordenadas (x, y)
- **Velocidad del avión:** vector velocidad (vx, vy)
- **Distancia al objetivo:** calculada cada frame
- **Sensores:** radar simple con alcance ilimitado

B) RAZONAMIENTO Estados del misil:

1. **BÚSQUEDA**
    - Condición: recién lanzado
    - Decisión: orientarse hacia posición actual del avión
2. **PERSECUCIÓN**
    - Condición: distancia > 200 px
    - Decisión: ajustar dirección hacia posición actual
3. **PREDICCIÓN**
    - Condición: distancia < 200 px
    - Decisión: calcular posición futura y dirigirse allí
4. **EXPLOSIÓN**
    - Condición: colisión detectada o tiempo agotado
    - Decisión: detonar

Cálculo de predicción:

cpp

```
// Estima dónde estará el jugador en T segundos

float leadTime = distancia / missileSpeed;

predictedX = playerX + playerVelX * leadTime * leadFactor;

predictedY = playerY + playerVelY * leadTime * leadFactor;
```

C) ACCIÓN

- Calcular vector dirección hacia objetivo predicho
- Normalizar vector
- Aplicar velocidad constante en esa dirección
- Actualizar rotación sprite del misil

D) APRENDIZAJE

- Cuenta número de fallos consecutivos
- Por cada 2 fallos: `leadFactor += 0.1`
- Mejora predicción con la experiencia

- leadFactor inicial = 1.0, máximo = 2.5