UNIVERSIDAD EAFIT
Organización de Computadores

Space Invaders

Implementación en Arquitectura Hack
Plataforma Nand2Tetris

Integrantes:
Andrés Felipe Vélez Álvarez
Simón Mazo Gómez
Sebastián Salazar Henao

Profesor:
Edison Valencia Díaz

19 de noviembre de 2025

Índice

1. Resumen Ejecutivo

Este documento presenta el desarrollo de una implementación del clásico videojuego arcade *Space Invaders* en lenguaje Jack para la arquitectura Hack (plataforma Nand2Tetris). El proyecto demuestra conceptos de organización de computadores, gestión de memoria y programación de bajo nivel.

El juego incluye sistema de vidas, niveles progresivos, puntuación diferenciada y mecánicas completas de disparo bidireccional.

2. Marco Teórico

2.1. Arquitectura Hack

Computadora de 16 bits diseñada para Nand2Tetris con:

- CPU RISC de 16 bits
- 32KB RAM, 32KB ROM
- Pantalla 512x256 píxeles monocromáticos (mapeo de memoria)
- Teclado con mapeo de memoria

2.2. Lenguaje Jack

Lenguaje orientado a objetos con sintaxis similar a Java/C++. Características:

- Gestión manual de memoria (sin garbage collector)
- Biblioteca estándar básica (Math, Array, Screen, Keyboard, etc.)
- Compilación a VM code → ASM → código máquina

3. Arquitectura del Software

3.1. Diseño Modular

El proyecto se estructura siguiendo principios de diseño orientado a objetos y separación de responsabilidades. La arquitectura consta de 8 clases principales:

Clase	**Responsabilidad**
Main	Punto de entrada del programa
Game	Controlador principal del juego
Player	Lógica de la nave del jugador
Fleet	Gestión de la flota de aliens
Alien16	Entidad individual de alien
AlienBomb	Proyectiles enemigos
Bullet	Proyectiles del jugador
Sprites	Biblioteca de gráficos bitmap

Cuadro 1: Módulos del sistema

3.2. Diagrama de Clases Conceptual

```
Main
  Game
       Player
           Bullet
       Fleet
           Alien16[]
           AlienBomb[]
       Sprites (biblioteca estática)
```

3.3. Descripción de Módulos Principales

Main: Punto de entrada que instancia y ejecuta el juego.

Game: Motor principal con loop a 100 FPS, gestión de estado (puntos, vidas, nivel), detección de colisiones y control de dificultad progresiva.

Player: Nave del jugador (32x32 px, velocidad 5px/frame, un proyectil activo).

Fleet: Gestiona 30 aliens en matriz 3×10 con movimiento sincronizado, detección de bordes y sistema de disparo aleatorio.

Alien16: Entidad individual con posición, estado vivo/muerto y valor en puntos (30/20/10 según fila).

Sprites: Biblioteca estática de gráficos bitmap (arrays de 16 enteros, cada uno representa una fila de 16 bits).

4. Proceso de Desarrollo

4.1. Fase 1: Diseño

Decisiones clave:

- **Gráficos:** Sistema bitmap 16x16 escalado 2x
- **Memoria:** Reutilización de objetos, sprites compartidos, liberación explícita
- **Colisiones:** Algoritmo AABB simplificado

4.2. Fase 2: Implementación Iterativa

Iteración 1: Prototipo básico (Player, movimiento, input).
Iteración 2: Flota con movimiento sincronizado y sistema de delay configurable.
Iteración 3: Sistema de combate completo (disparos, colisiones, puntuación).
Iteración 4: Niveles, vidas y progresión de dificultad.
Iteración 5: Interfaz (pantallas de inicio, puntuación, HUD, game over).

5. Problemas Encontrados y Soluciones

5.1. Problema 1: Heap Overflow

Error: `ERR: Heap overflow (code 6)` al cambiar de nivel.

Causa: Sprites temporales no liberados en `showScoreTable()`.
Solución: Liberar memoria explícitamente con `dispose()` después de usar arrays temporales.

5.2. Problema 2: Illegal Pixel Coordinates

Error: `ERR: Illegal pixel coordinates (code 7)` esporádicamente.
Causa: Coordenadas inválidas (¡0 o ¿límites de pantalla) en funciones de dibujo.
Solución: Validación estricta de rangos antes de cada `Screen.drawRectangle()`:

```
1 if (x < 0 | y < 0 | x2 > 511 | y2 > 255) { return; }
```

5.3. Problema 3: Velocidad Inconsistente

Causa: Delay fijo sin compensación por carga computacional variable.
Solución: Sistema de ticks con delay adaptativo para movimiento de aliens independiente del framerate.

6. Características Implementadas

6.1. Mecánicas de Juego

6.1.1. Sistema de Puntuación Diferenciada

Tipo de Alien	Puntos	Posición
Alien Tipo A	30	Fila superior
Alien Tipo D	20	Fila media
Alien Tipo B	10	Fila inferior

Cuadro 2: Sistema de puntuación

6.1.2. Sistema de Vidas

- El jugador inicia con 3 vidas
- Se pierde una vida cuando una bomba alien impacta la nave
- Se pierde una vida cuando un alien alcanza y=220 (línea de defensa)
- Game over al agotar todas las vidas

6.1.3. Progresión de Niveles

Al eliminar los 30 aliens:

- Incremento de nivel
- Regeneración de flota completa
- Reducción de `fleetMoveDelay` en 2 unidades

- Velocidad máxima al llegar a delay = 4

Fórmula de dificultad:

$$\text{fleetMoveDelay} = \text{máx}(4, 20 - 2 \times (\text{level} - 1)) \quad (1)$$

6.2. Interfaz de Usuario

6.2.1. Pantalla de Inicio

- Título centrado del juego
- Instrucciones de control
- Prompt para continuar a tabla de puntuación

6.2.2. Tabla de Puntuación

- Visualización de sprites de cada tipo de alien
- Puntos asociados a cada tipo
- Información del sistema de vidas
- Transición al juego principal

6.2.3. HUD en Tiempo Real

Información mostrada continuamente:

```
PUNTOS:150 VIDAS:2 NIVEL:3 ALIENS:12
```

6.2.4. Pantalla de Game Over

- Mensaje de fin de partida
- Nivel alcanzado
- Puntuación final
- Mensaje de despedida

7. Optimizaciones Implementadas

Rendering: Sistema clear-and-draw local (solo área del sprite) para evitar parpadeo.

Colisiones: Early termination al detectar colisión, skip de aliens muertos.

Memoria: Sprites compartidos entre aliens del mismo tipo (reducción de 960 a 128 bytes).

8. Pruebas y Métricas

8.1. Casos de Prueba Principales

- Movimiento y límites de pantalla (izquierda/derecha, bordes)
- Sistema de disparo (creación, cooldown, límite de un proyectil activo)
- Colisiones (bala-alien, bomba-jugador)
- Progresión (eliminación de flota, cambio de nivel, game over)

8.2. Métricas de Rendimiento

Métrica	**Valor**
Framerate objetivo	100 FPS (10ms/frame)
Delay movimiento inicial	200ms
Delay mínimo (nivel avanzado)	40ms
Cooldown de disparo	150ms
Aliens simultáneos	30
Bombas simultáneas	10 (máximo)

Cuadro 3: Métricas de rendimiento

9. Guía de Usuario

9.1. Instalación

1. Descargar herramientas Nand2Tetris desde `https://www.nand2tetris.org/software`
2. Clonar repositorio: `git clone https://github.com/Salazar1022/Space_Invaders_OC.git`
3. Compilar: `.\scripts\build_and_run.ps1 'C:\nand2tetris\tools'`

9.2. Controles

- **Flecha izquierda:** Mover nave a la izquierda
- **Flecha derecha:** Mover nave a la derecha
- **Barra espaciadora:** Disparar

9.3. Objetivo

Eliminar todos los aliens sin perder las 3 vidas. Aliens superiores valen 30 puntos, medios 20, inferiores 10. Evitar bombas enemigas y que los aliens lleguen a la línea de defensa.

10. Conclusiones

10.1. Logros

1. Videojuego completo y funcional en arquitectura Hack con recursos limitados
2. Optimización efectiva mediante gestión manual de memoria y compartición de recursos
3. Diseño modular en 8 clases independientes
4. Comprensión profunda de arquitectura de computadores y programación de bajo nivel

10.2. Desafíos Superados

- Gestión manual de memoria sin garbage collector
- Implementación de sprites sin bibliotecas gráficas
- Optimización de rendimiento en operaciones de pantalla
- Debugging en entorno de bajo nivel

10.3. Aprendizajes Clave

Técnicos: Gestión de memoria, programación de bajo nivel, manipulación de bits, optimización de rendimiento.

Ingeniería: Diseño iterativo, debugging sistemático, separación de responsabilidades, documentación efectiva.

Conceptuales: Abstracción en capas, creatividad bajo restricciones, importancia del testing exhaustivo.

Aplicables a: Sistemas embebidos, optimización de software, desarrollo de videojuegos, sistemas operativos, compiladores.

11. Referencias

1. Nisan, N., & Schocken, S. (2005). *The Elements of Computing Systems: Building a Modern Computer from First Principles*. MIT Press.
2. Nand2Tetris Official Website: `https://www.nand2tetris.org`
3. Jack Language Specification: `https://www.nand2tetris.org/project09`
4. Hack Computer Specification: `https://www.nand2tetris.org/project05`
5. Space Invaders Wikipedia: `https://en.wikipedia.org/wiki/Space_Invaders`
6. Repositorio del proyecto: `https://github.com/Salazar1022/Space_Invaders_OC`

12. Apéndices

12.1. Apéndice A: Estructura de Archivos

```
Space_Invaders_OC/
 README.md, QUICKSTART.md, documentacion.tex
 scripts/ (build_and_run.ps1, download_nand2tetris.ps1)
 src/ (Main.jack, Game.jack, Player.jack, Fleet.jack,
          Alien16.jack, AlienBomb.jack, Bullet.jack,
          Sprites.jack, *.vm generados)
```

12.2. Apéndice B: Algoritmo de Colisiones AABB

```
method boolean collidesWith(int x1, int y1, int x2, int y2) {
    var int myX2, myY2;
    let myX2 = x + 31;
    let myY2 = y + 31;

    if (x > x2) { return false; }
    if (myX2 < x1) { return false; }
    if (y > y2) { return false; }
    if (myY2 < y1) { return false; }

    return true;
}
```

12.3. Apéndice C: Comandos de Compilación

```
# Jack -> VM
java -jar C:\nand2tetris\tools\JackCompiler.jar src

# Ejecutar en VM Emulator
java -jar C:\nand2tetris\tools\VMEmulator.jar
# File > Load Program > src, luego Run (F5)
```