Universidad Nacional Sede Regional Brunca

Campus Pérez Zeledón / Campus Coto
Arquitectura de computadores
II Proyecto - II ciclo 2025

Prof: M.C. Gabriel Núñez M.

Valor: 14 %

Objetivo

Desarrollar un juego de exploración en ensamblador 8086/8088 que implemente un mundo abierto en modo gráfico, cargado desde archivo, con un sistema de renderizado por viewport, recolección de recursos y gestión de inventario, aplicando los conceptos de programación a bajo nivel y arquitectura de computadores.

Instrucciones iniciales

1. Lea y comprenda cuidadosamente lo que se le solicita.
2. El proyecto podrá ser realizado en grupos de máximo 2 o 3 personas. Para grupos de 3 personas se deberán implementar funcionalidades adicionales.
3. El proyecto debe ser desarrollado completamente en lenguaje ensamblador para arquitectura 8086/8088.
4. El juego debe utilizar modo gráfico.
5. Si se comprueba que existen dos o más proyectos similares se procederá a colocar nota cero a todos los proyectos involucrados.
6. El proyecto debe ser subido al Aula Virtual en la fecha indicada, incluyendo todos los archivos fuente (.ASM), archivos de mapa y documentación requerida.
7. Cada rutina del programa debe estar debidamente comentada y organizada modularmente.
8. El proyecto deberá ser defendido en el día indicado por todos los miembros del grupo.
9. Utilizar el emulador DOSBox (https://www.dosbox.com/download.php?main=1) o similar para el desarrollo y pruebas.
10. Utilizar la Documentación oficial según Libro y HelpPC (https://helppc.netcore2k.net/topics)
11. Fechas importantes:
    - Entrega del enunciado: 15 al 19 de septiembre del 2025.
    - Entrega del avance: 13 al 17 de octubre del 2025.
    - Entrega del proyecto: 03 al 07 de noviembre del 2025.

Inspiración y Fundamentos Conceptuales
El proyecto se basa en conceptos de juegos de exploración clásicos con gráficos pixelados, combinados con técnicas de programación a bajo nivel:

- Juegos de aventura textual: Como los clásicos "Zork" o "Colossal Cave Adventure", donde la exploración se realiza mediante descripciones textuales.
- Juegos roguelike: Como "NetHack" o "ADOM", que utilizan representación en caracteres ASCII para crear mundos complejos.
- Juegos de aventura clásicos: Como "King's Quest" o "The Legend of Zelda", que utilizaban modos gráficos limitados pero creativos.
- Manipulación directa de hardware: Acceso a los puertos de video y controlador de gráficos.
- Sistemas de archivos: Implementación de persistencia de datos mediante manipulación directa de archivos.
- Gráficos por computadora: Conceptos de memoria de video, planos de bits y paletas de colores. Conceptos básicos de renderizado y gestión de memoria de video.

Especificaciones Técnicas y Requisitos
Los estudiantes deberán desarrollar un sistema que integre los siguientes aspectos clave:

1. Sistema de carga de mapas y recursos gráficos
   - Implementar lectura de archivos que definan el mundo y sus elementos
   - Carga de sprites o tilesets para representar elementos gráficos
   - Gestión de paleta de colores
2. Renderizado gráfico por viewport
   - Visualizar sólo la porción del mapa alrededor del jugador (viewport)
   - Dibujado eficiente de tiles/sprites en modo gráfico
   - Actualización parcial de pantalla para optimizar rendimiento
     item Scroll suave al moverse por el mundo
3. Sistema de movimiento y exploración
   - Control del personaje mediante teclado (teclas de dirección o WASD)
   - Animación básica del personaje al moverse
   - Detección de colisiones con obstáculos
4. Mecánica de recolección de recursos
   - Diferentes tipos de recursos representados gráficamente
   - Sistema de inventario visual para almacenar recursos
   - Interfaz gráfica para mostrar recursos obtenidos

5. Especificación de recursos
    - Mínimo 3 tipos diferentes de recursos recolectables
    - Cada recurso debe tener representación visual única (sprite diferente)
    - Sistema de inventario con capacidad para 8 tipos de items máximo
    - Los recursos deben aparecer en ubicaciones específicas del mapa
    - Debe existir al menos 5 instancias de cada recurso en el mapa
6. Interfaz de usuario gráfica
    - HUD (Head-Up Display) con información del juego
    - Indicadores de progreso y objetivos
    - Menús e interfaces visuales
7. Sistema de colisiones y física
    - Detección de colisiones pixel-perfect o por bounding boxes
    - Mínimo 3 tipos diferentes de terrenos con propiedades distintas
    - Implementación de obstáculos no transitables
    - Límites del mapa bien definidos
8. Objetivos y sistema de progresión
    - Objetivo principal: recolectar al menos 2 unidades de cada tipo de recurso
    - Contador visible de recursos recolectados
    - Indicador de progreso hacia la finalización del juego
    - Mensaje de victoria al completar todos los objetivos

En caso de grupos de 3 personas
Se deberán implementar las siguientes características adicionales:

- Sistema de guardado y carga de partidas
- Múltiples niveles o mapas con transiciones entre ellos
- Animaciones más complejas para personaje y elementos del juego
- Sistema de enemigos o NPCs básicos con IA simple
- Implementación de efectos de sonido mediante el speaker del PC
- Sistema de diálogos o interacciones con NPCs
- Menú de inicio y pantalla de game over mejoradas

Formatos de archivo
Los mapas y recursos deberán seguir formatos definidos:

```
Formato de archivo de mapa:
[Ancho] [Alto]
[Datos del mapa (una matriz de Ancho x Alto)]
[R] [X] [Y] [Tipo] [Cantidad]  ; Recursos
; Donde:
; - Ancho, Alto: dimensiones del mapa (máx. 100x100)
; - Datos del mapa: valores numéricos (0-255) representando tiles
; - R: indica recurso, X/Y: coordenadas, Tipo: 1-3, Cantidad: 1-5

Formato de archivo de sprites:
[Ancho] [Alto]
[Datos de píxeles] (valores de color 0-F)
; Ancho y Alto recomendados: múltiplos de 8 (ej. 16x16, 32x32)
```

Requisitos Técnicos Específicos

- Utilizar modo gráfico EGA (640x350, 16 colores)
- Los 16 colores disponibles son los estándar:
  - 0: Negro
  - 1: Azul
  - 2: Verde
  - 3: Cyan
  - 4: Rojo
  - 5: Magenta
  - 6: Marrón
  - 7: Blanco
  - 8: Gris oscuro
  - 9: Azul claro
  - 10: Verde claro
  - 11: Cyan claro
  - 12: Rojo claro
  - 13: Magenta claro
  - 14: Amarillo
  - 15: Blanco brillante
- Gestionar adecuadamente la paleta de colores EGA
- Considerar las limitaciones de color al diseñar los sprites
- Tamaño mínimo del viewport: 160x100 píxeles
- El personaje debe estar siempre centrado en el viewport
- Los sprites deben tener tamaño máximo de 32x32 píxeles
- Implementar técnica de doble búfer para evitar parpadeo:
  - Dibujar primero en una zona de memoria oculta
  - Mostrar la imagen completa solo cuando esté terminada
  - Esto crea una transición suave entre frames
- El mapa debe tener tamaño mínimo de 20x15 tiles
- Cada tile debe ser de 16x16 píxeles como mínimo
- Manejar al menos 4 direcciones de movimiento (arriba, abajo, izquierda, derecha)

Criterios de Evaluación

1. Funcionalidad (40 %):
   - Correcta carga y renderizado de mapas (10 %)
   - Sistema de movimiento y colisiones funcional (10 %)
   - Mecánica de recolección de recursos implementada (10 %)
   - Interfaz de usuario gráfica clara y responsiva (5 %)
   - Finalización del juego con condiciones de victoria (5 %)
2. Calidad del código (25 %):
   - Organización modular y comentarios adecuados (10 %)
   - Uso eficiente de interrupciones y servicios del DOS (5 %)
   - Optimización de memoria y gestión de recursos (5 %)
   - Manejo adecuado de errores y casos límite (5 %)
3. Calidad gráfica y visual (15 %):
   - Coherencia visual y estética (5 %)
   - Animaciones fluidas del personaje (5 %)
   - Diseño de sprites y elementos visuales (5 %)
4. Experiencia de juego (10 %):
   - Jugabilidad fluida e intuitiva (5 %)
   - Progresión bien balanceada (5 %)
5. Exposición y defensa (10 %):
   - Claridad en la explicación (5 %)
   - Dominio técnico demostrado (5 %)

Recursos y Herramientas Recomendadas

- Emulador: DOSBox
- Ensamblador: TASM
- Editores: Notepad++, Visual Studio Code o similar
- Depuración: Debug de DOS
- Utilidades gráficas: Herramientas para crear y convertir sprites (ej. ProMotion NG, Aseprite)

Notas importantes

- Se recomienda comenzar con un motor gráfico simple e ir expandiendo funcionalidades
- La optimización del código es crucial para lograr un rendimiento aceptable en 8086/8088
- Considerar las limitaciones de memoria al diseñar los mapas y recursos gráficos