

Proyecto I – Invaders

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Segundo Semestre 2017
Valor 25%



Objetivo General

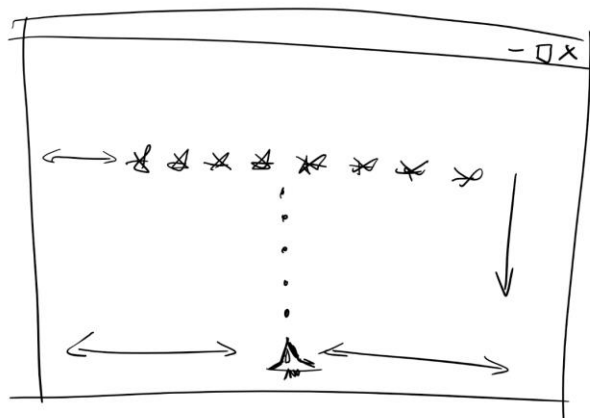
- Diseñar e implementar un juego al estilo de Space Invaders, que utilice listas enlazadas en distintas secciones del juego.

Objetivos Específicos

- Implementar listas enlazadas y algunas variaciones
- Investigar y desarrollar una aplicación en el lenguaje de programación Java
- Investigar acerca de programación orientada a objetos en Java.
- Aplicar patrones de diseño en la solución de un problema.

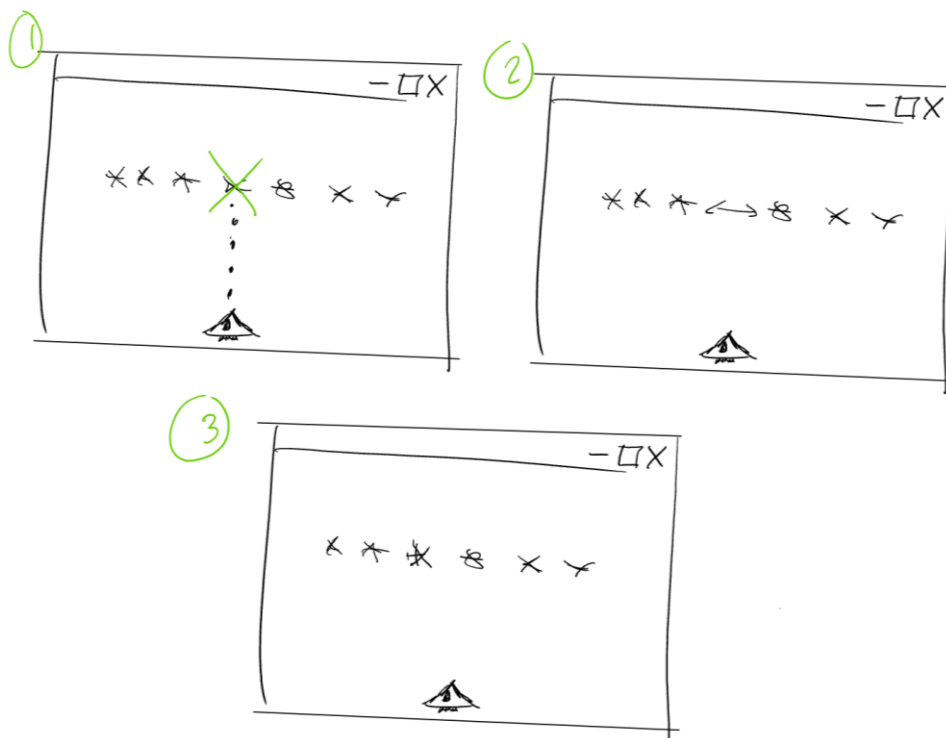
Descripción del Problema

Invaders es un juego en el que el jugador puede controlar una nave que se mueve horizontalmente en la parte inferior de la pantalla con el fin de eliminar a todos los invasores. Los invasores son hileras de enemigos que avanzan de arriba hacia abajo en la pantalla y se mueven de lado a lado. (el movimiento es de la hilera completa). El jugador deberá mover la nave para eliminarlos y evitar que lleguen hacia la parte inferior de la pantalla.



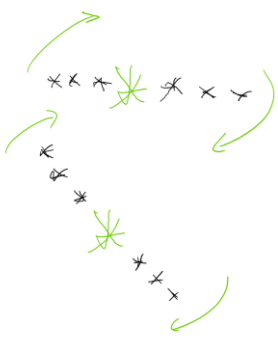
Las hileras de enemigos varían aleatoriamente. Cualquier tipo de hilera de enemigos se implementa mediante una lista enlazada y poseen ciertos atributos especiales. Conforme los niveles avanzan, los enemigos son más rápidos.

Conforme el jugador destruye los enemigos, los miembros de la hilera se mueven hacia el centro:



Existen las siguientes clases de hileras de enemigos:

Basic	Todos los enemigos de la hilera son iguales.	
Clase A	Uno de los enemigos de la hilera es jefe. Aleatoriamente se elige a un jefe. Cuando el jefe se destruye, el resto de naves desaparece. Para destruir el jefe, se requieren entre 2 a 5 disparos	
Clase B	Clase A con una variación. El jefe se intercambia rápidamente con el resto de los miembros de la hilera. Es una lista doblemente enlazada.	
Clase C	Igual que la A pero si el jefe se destruye, otro toma su lugar aleatoriamente. Es una lista circular.	

Clase D	Igual que la C pero los miembros de la hilera tienen diferentes resistencias. La Hilera se mantiene ordenada de mayor a menor resistencia, utilizando bubble sort.	
Clase E	Igual que la C con algunas variaciones. La hilera no es horizontal, constantemente rota según las manecillas del reloj. El jefe siempre está en el centro. Es una lista doble y circular.	

En alguna parte de la pantalla se debe indicar el tipo de hilera actual y el próximo por salir. Se muestra el nivel actual y la cantidad de puntos que lleva el usuario así como cualquier otra información relevante.

El jugador controla la nave de dos formas:

- Mediante el teclado
- Mediante una sencilla aplicación en el celular que utilizar el acelerómetro para mover la nave de lado a lado. La aplicación muestra también el tipo de hilera actual y el próximo por salir. Se muestra el nivel actual y la cantidad de puntos que lleva el usuario así como cualquier otra información relevante.

Documentación requerida

1. Se deberá documentar el código fuente utilizando JavaDoc.
2. Se deberá entregar un documento que contenga:
 - a. Todas las partes estándar: Portada, índice, introducción, conclusión, bibliografía y anexos.
 - b. El cuerpo del documento debe contener:
 - Breve descripción del problema
 - **Planificación y administración del proyecto**
 - ◆ Lista de features e historias de usuario identificados de la especificación
 - ◆ Distribución de historias de usuario por criticalidad y secuencia de uso
 - ◆ Minimal system span
 - ◆ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
 - ◆ Descomposición de cada user story en tareas.
 - **Diseño**
 - ◆ Diagrama de componentes
 - ◆ Diagrama de arquitectura
 - ◆ Diagrama de secuencia (Seleccionar al menos 4 Historias de Usuario)
 - ◆ Diagrama de clases inicial
 - ◆ Diagrama de clases final (generado utilizando el IDE)

→ Implementación

- ◆ Descripción de las estructuras de datos desarrolladas.
- ◆ Descripción detallada de los algoritmos desarrollados.
- ◆ Problemas encontrados: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo. Incluye descripción detallada, intentos de solución sin éxito, solución encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso
3. El trabajo es **individual**.
4. Es obligatorio utilizar un manejador de versiones del código. Puede ser git o svn. Se revisará que cada commit lleve comentarios relevantes relacionados con alguna tarea identificada en la sección de planificación
5. Los proyectos que no cumplan con los siguientes requisitos **no serán revisados**:
 - a. Toda la solución debe estar integrada
6. El código tendrá un valor total de 65%, la documentación 30% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
10. Se debe enviar el código (preliminar) y la documentación a más tardar a las 23:59 del día de la fecha de entrega al email del profesor. **Se debe seguir el formato del Subject descrito en el Programa del Curso.**
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial, momento en el cual deben enviar un email con el código fuente, si no lo hacen se asumirá que la versión oficial del código fue la enviada el día de la fecha de entrega junto con la documentación. **Se debe enviar en el mismo correo del punto 8.**
13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
 - a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
 - b. Si no se utiliza un manejador de código se obtiene una nota de 0.
 - c. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
 - d. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
 - e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
 - f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
 - g. La nota de la documentación debe ser acorde a la completitud del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.

15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.