Test Plan

Tipo de prueba	Naturaleza de la prueba	Ejemplo
Al empezar el programa el menú debera mostrarse correctamente y ser funcional	Checar la visualización del menú y el menú debería de responder según la elección del usuario	“Jugar” “Opciones” “Salir” El usuario presiona jugar y comienza el juego
Checar que los procesos que se lleven a cabo sean trabajos en diferentes threads	En el código fuente modificarlo para que imprima el nombre del thread actual	**Código** System.out.println("Thread id de window: "+w.getId()+" nombre: "+w.getName()); System.out.println("Thread id de Player Manager: "+PMan.getId()+" nombre: "+PMan.getName()); **Salida** Thread id de window: 34 nombre: Thread-17 Thread id de Player Manager: 36 nombre: Thread-19
Checar la entrada de inputs a través del teclado	El usuario utiliza las flechas del teclado para mover el jugador	El usuario presiona la tecla derecha y se moverá hacia la derecha el jugador
Checar que haya movimientos de gráficos y manejo correctamente de estos.	El usuario moverá el jugador y tanto el personaje del usuario como del enemigo cambiaran su skin conforme caminen	El usuario se mueve a la derecha y la skin del jugador y el enemigo se mueven para imitar el caminar hacia la derecha
Checar la facilidad de uso del programa.	Se le dará a jugar a gente que no sepa sobre el juego con una breve explicación	Se le dirá al tester los controles y el objetivo del juego.

		“Te mueves con las flechas y atacas con espacio. Trata de sobrevivir”
Se checará la funcionalidad de guardar el progreso	Se salvará el juego en algún punto durante una prueba. Se cerrará y se espera que se reanude en el punto guardado.	Se guarda en juego con un nivel 5 y 30 puntos de vida. Cuando se vuelve a reanudar el juego los datos se mantienen iguales.
Se verifica las colisiones entre objetos dentro del programa	El usuario tratará de salirse de los límites del programa o atravesar obstáculos.	Al tratar de atravesar los límites, el jugador deberá de dejar de moverse.
Se checa el manejo de la Salud en el programa	Tanto el enemigo como el jugador recibirán daño hasta que alguno muera	Al recibir daño la salud debe de bajarse acorde al nivel del jugador.
Se verá el impacto del nivel en los ataques y salud	Se verá que se modifique la salud y la vida acorde al nivel del usuario	Al nivel 5 el ataque del jugador deberá ser mayor y proporcional al ataque que tenía en nivel 1
Se verificará al ratio en el que los enemigos se crean dependiendo del nivel	En cada nivel se buscará sacar un promedio donde este esté dentro de los valores esperados para cada nivel	Al nivel 5 deberán de aparecer 2 enemigos cada 10 segundos.

Diagrama Caso de Uso

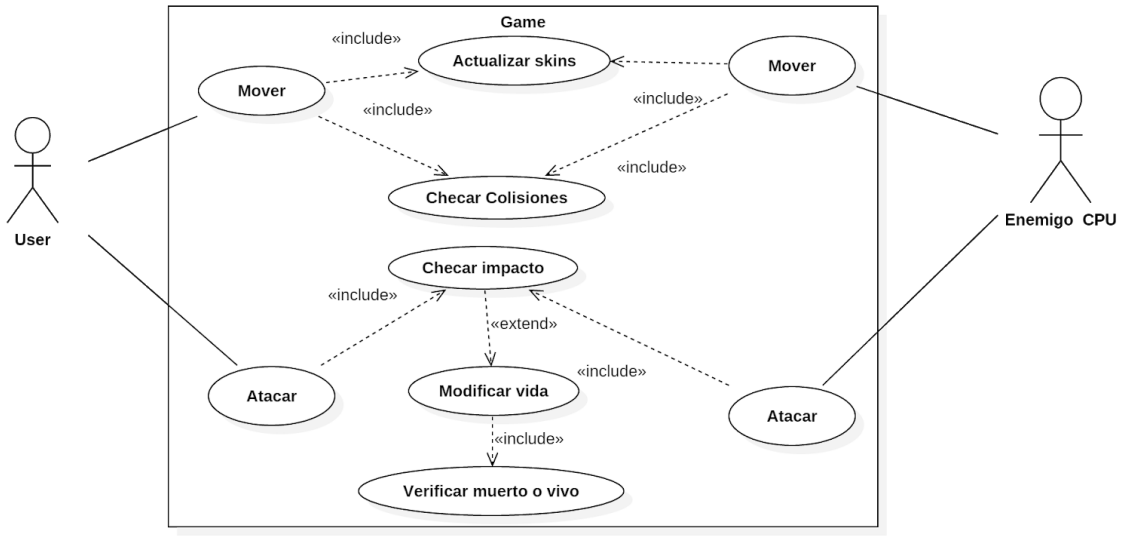


Diagrama Máquina de estados finitos (Finite-State Machine) para el comportamiento de los enemigos

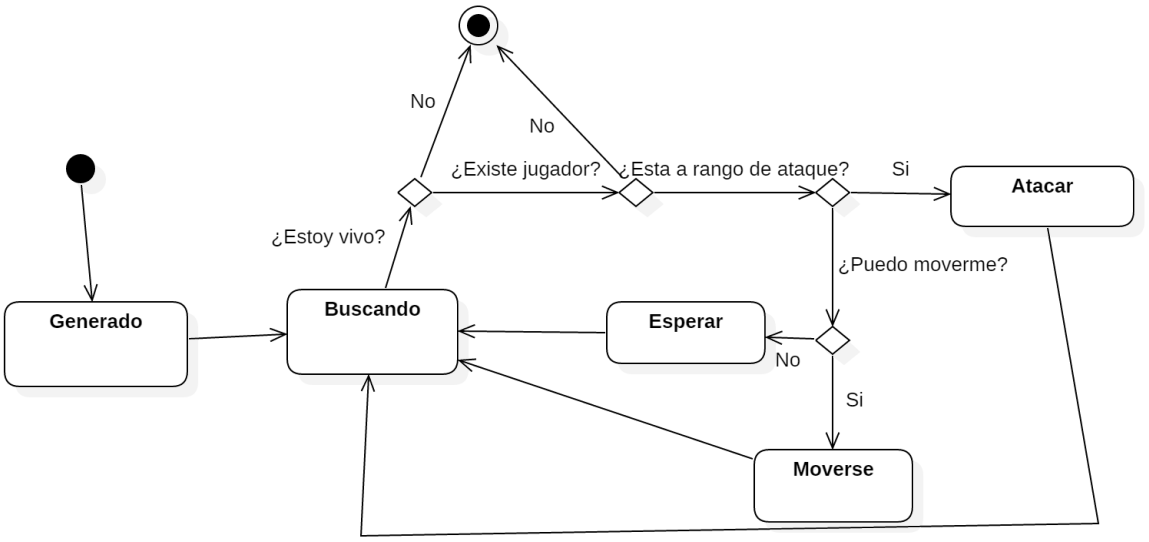
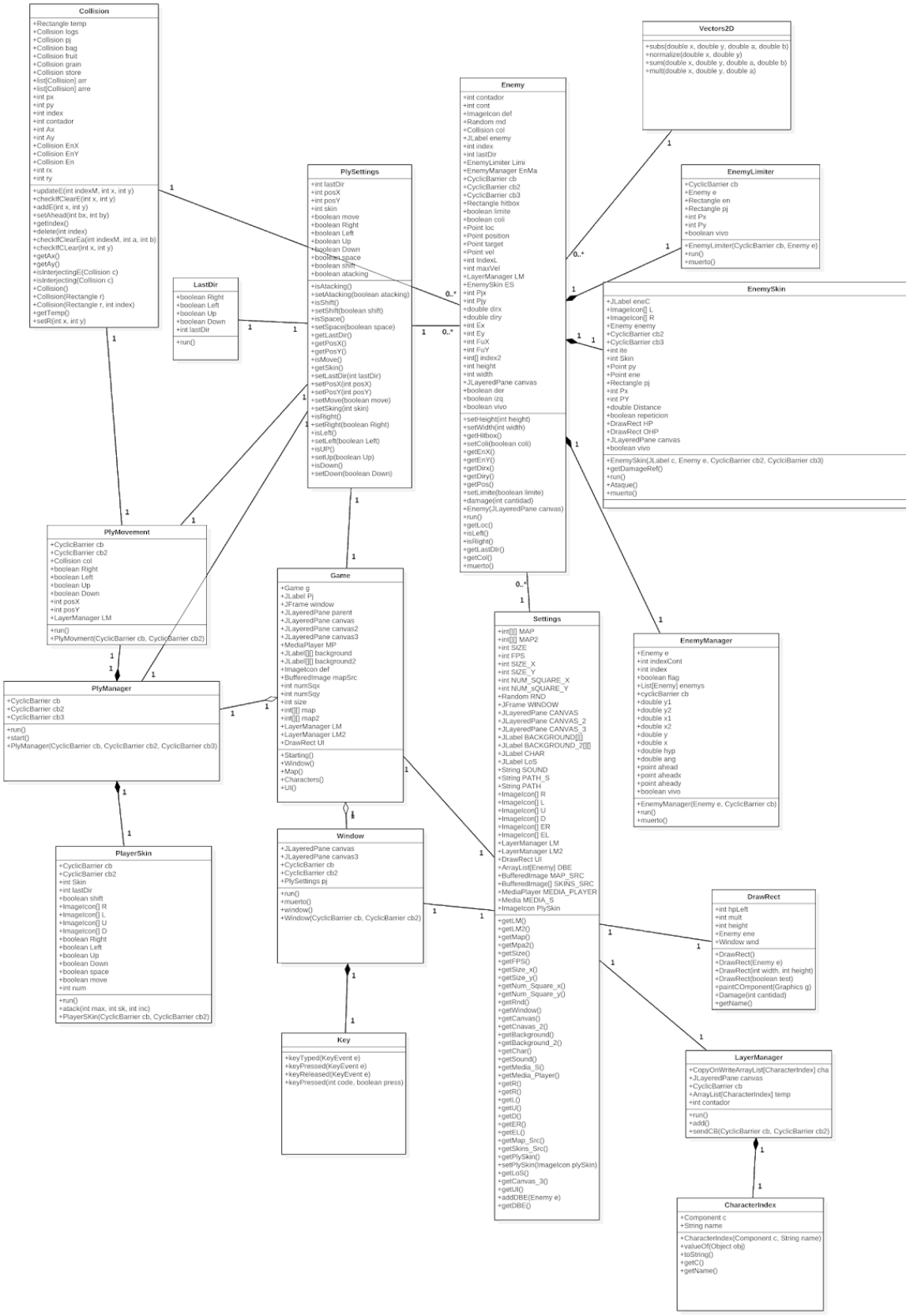


Diagrama de Clases (Simplificado)

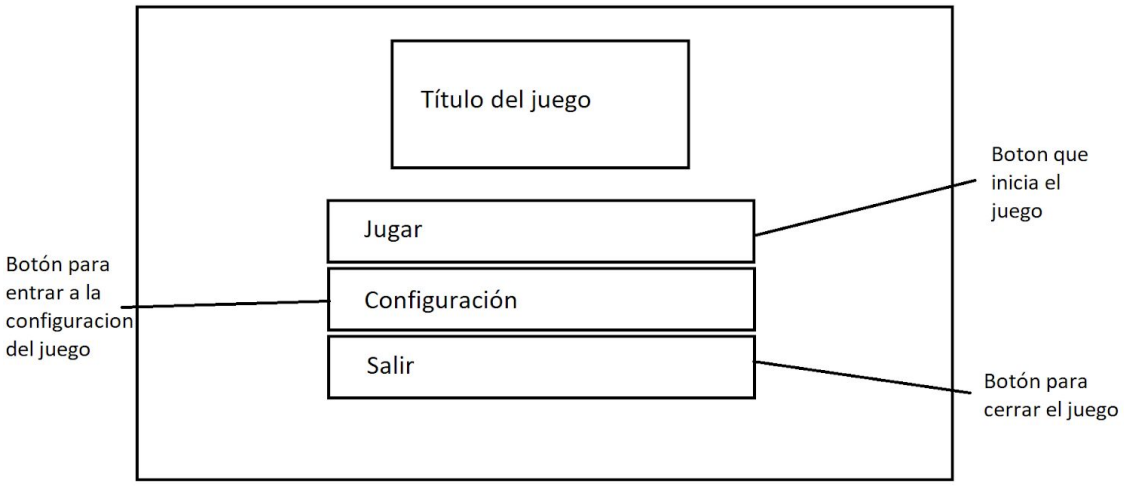


Funcionalidad de cada clase:

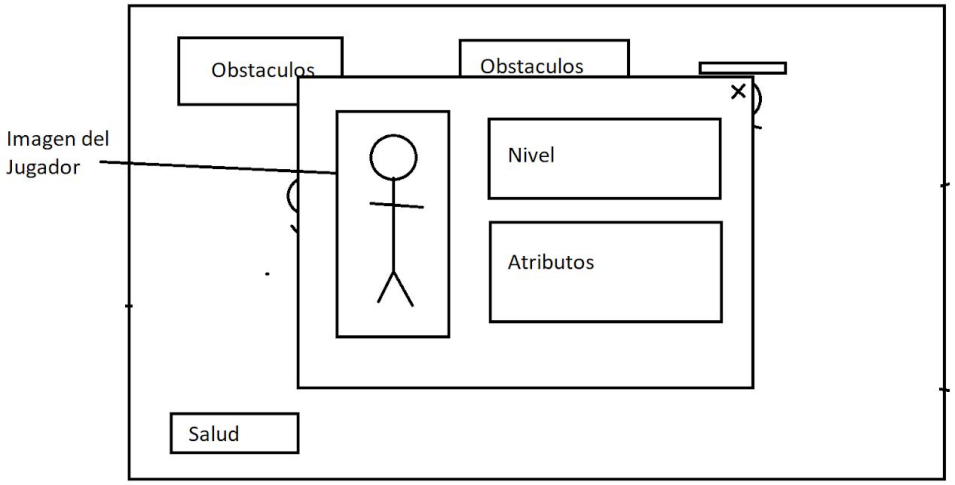
Window - Se encarga de actualizar los gráficos
Game - Se encarga de inicializar todo el juego e iniciar los diferentes threads.
PlyManager - Se encarga de mantener sincronizados los threads de PlyMovement y PlayerSkin
PlayerSkin - Se encarga de actualizar la Skin del jugador
PlyMovement - Controla el movimiento del jugador
Key - Maneja la entrada del teclado
Settings - Guarda toda la configuración y variables que se manejan en todo el programa
PlySettings - Guarda la información relacionada con las variables del jugador
DrawRect - Maneja la salud de los personajes
LayerManager - Maneja la profundidad de los personajes dependiendo del eje y
CharacterIndex - Mantiene la lista de los personajes que están en la pantalla
EnemyManager - Mantiene la sincronización entre EnemySkin, Enemy y EnemyLimiter.
EnemySkin - Maneja la skin de los personajes enemigos
EnemyLimiter - maneja las colisiones del enemigo
Vectors2D - Implementa operaciones para puntos 2D
Enemy - Se encarga del movimiento y el ataque del personaje enemigo
LastDir - Mantiene control de la última dirección tomada por el jugador
Collision - Se encarga de mantener y controlar las colisiones entre obstáculos y personajes

Diseño e Interfaz:

Menú



Menú durante el juego



Recursos y diseño de personajes y mapa

Recurso del mapa

Sprites para enemigos (Ataque Izquierda)

Sprites para Personaje (Ataque Izquierda)

Diagrama de flujo para Movimiento de los enemigos

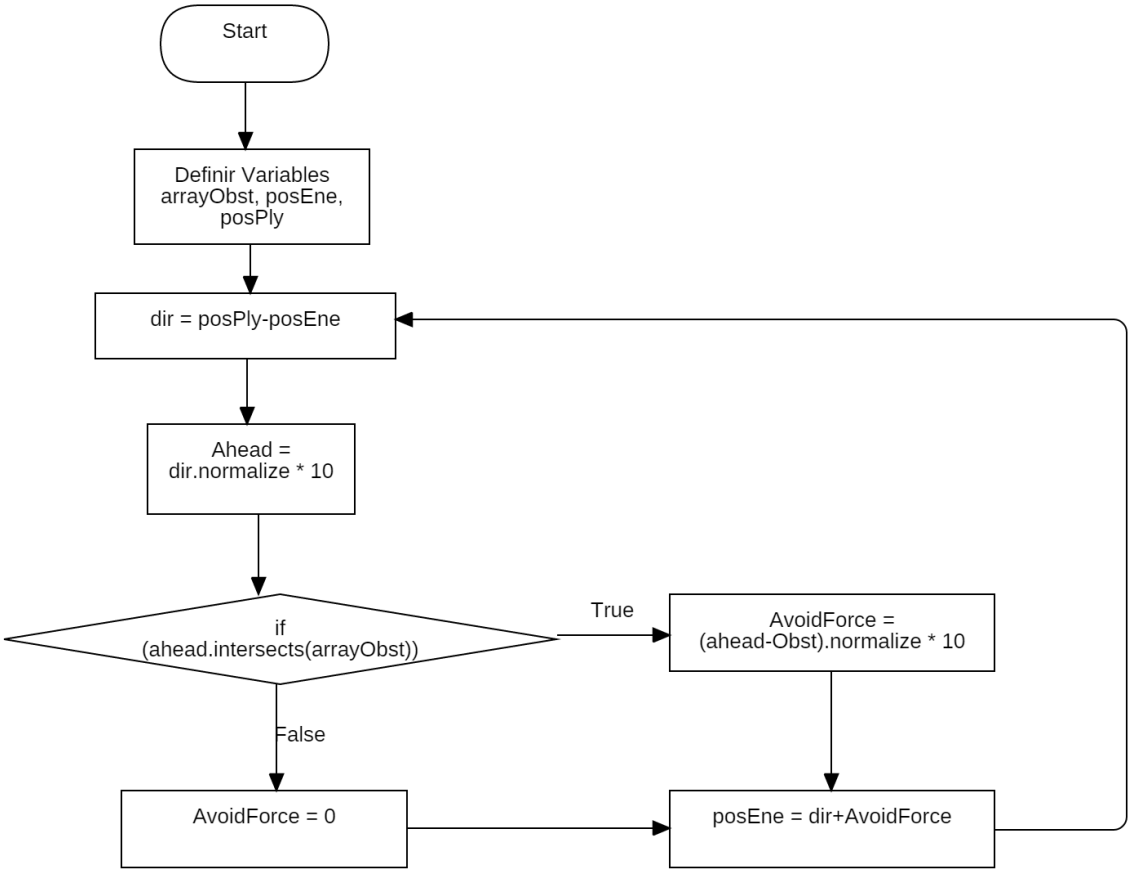


Diagrama de flujo Ataque para el enemigo

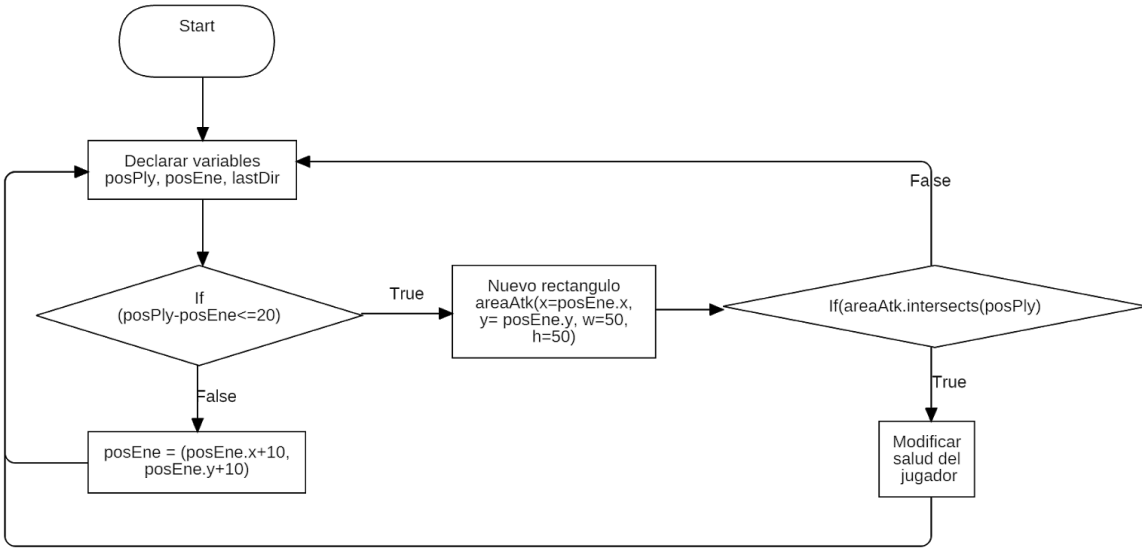
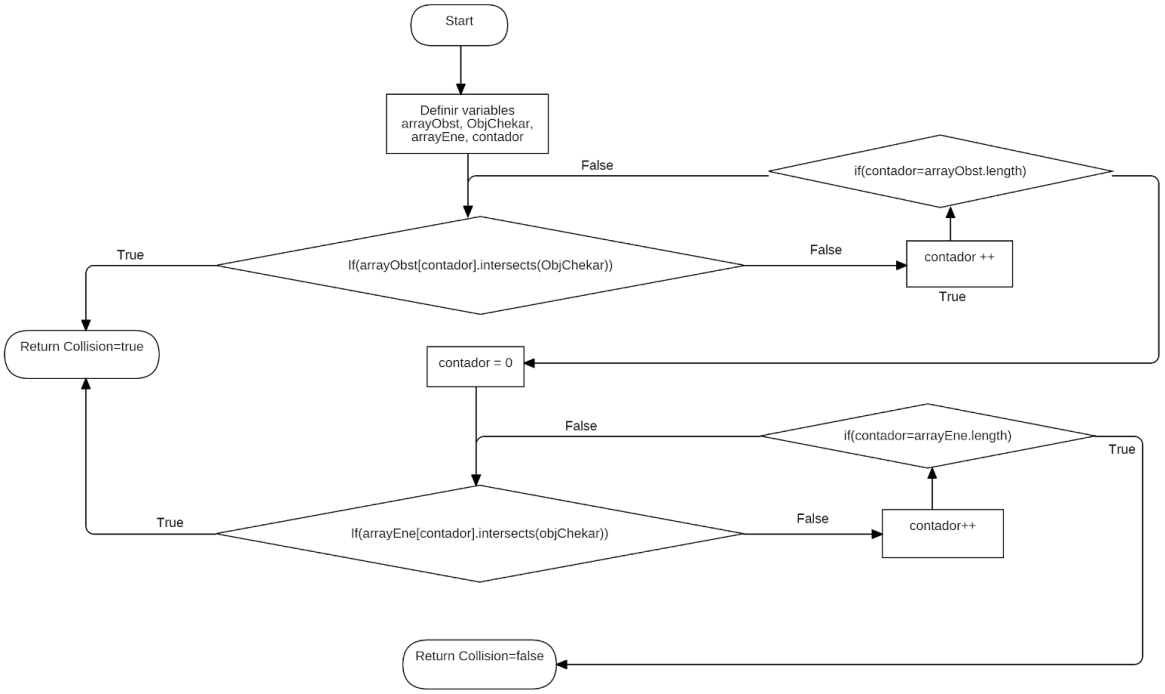


Diagrama de flujo para colisiones entre personajes y obstáculos


Start
Definir variables arrayObst, ObjChekar, arrayEne, contador
if(contador=arrayObst.length)
False
True
If(arrayObst[contador].intersects(ObjChekar))
True
False
contador ++
True
Return Collision=true
contador = 0
if(contador=arrayEne.length)
False
True
If(arrayEne[contador].intersects(objChekar))
True
False
contador++
Return Collision=false