**Logotipo

Descripción generada automáticamente**

**Documentación del proyecto final informática II**

Juan Camilo Peñaloza Rodríguez

Cristian Castaño Herrera

Universidad de Antioquia  
Facultad de Ingeniería

Ingeniería Electrónica y de Telecomunicaciones

Medellín, Antioquia, Colombia

**A la deriva**

**Descripción**

"A la deriva: El Viaje de las Hojas" es un juego de aventuras en el que el jugador controla una hoja que ha sido arrancada de su árbol por una ráfaga de viento. La duración del juego se basa en el tiempo que tarda la hoja en encontrar un nuevo hogar seguro antes de que sea destruida por los elementos climáticos. El juego tiene lugar en un mundo vibrante y colorido donde el clima es un factor importante. La hoja debe sobrevivir a los diferentes tipos de clima que encuentra a lo largo de su viaje, incluyendo lluvia, nieve, viento y sol abrasador. Cada tipo de clima tiene sus propias físicas, desafíos y obstáculos el jugador debe utilizar habilidades especiales de la hoja, como la capacidad de flotar y de doblarse, para superarlos.

**Mecánica del Juego**:

El objetivo principal del juego es encontrar un hogar seguro para la hoja de papel antes de que sea destruida por los elementos climáticos. El jugador controla la hoja y debe utilizar habilidades especiales para superar los desafíos y obstáculos que se presentan en el camino. La hoja puede flotar y doblarse, lo que le permite navegar por el entorno de manera más eficiente.

Los elementos climáticos mencionados cumplen un rol muy importante dentro del juego. La hoja debe sobrevivir a diferentes tipos de clima, como lluvia, granizo, nieve, viento y sol abrasador. Cada tipo de clima tiene sus propias físicas y desafíos y obstáculos únicos, y el jugador debe adaptarse y utilizar habilidades especiales para superarlos.

El juego se divide en niveles. Cada nivel presenta un entorno diferente con desafíos y obstáculos únicos. El jugador tiene un tiempo limitado para explorar cada nivel y encontrar un hogar seguro para la hoja antes de que sea destruida por los elementos climáticos.

El juego también incluirá un sistema de puntuación que recompensará al jugador por encontrar lugares seguros más rápido, lo que aumentará su puntuación total. También habrá objetos coleccionables, como piezas de papel y plumas, que el jugador puede recolectar para desbloquear habilidades especiales y otros extras.

**Físicas:**

El juego "A la deriva" incorporará diversas físicas para proporcionar una experiencia de juego más realista y emocionante. Algunas de las físicas que se incluirán son:

- Caída libre: Cuando la hoja cae de un lugar alto o es llevada por el viento hacia abajo, seguirá las leyes de la caída libre para simular la gravedad. Además, también algunos de los elementos climáticos como lo son: la lluvia, la nieve y el granizo también seguirán las mismas leyes.

- MRU (Movimiento rectilíneo uniforme): La hoja se moverá con una velocidad constante en línea recta en algunos momentos del juego. Habrá también otros personales tales como pájaros que se moverán de la misma forma en dirección contraria a la que va la hoja, para así dar la sensación de que la hoja y el pájaro se pueden chocar.

- MRUA (Movimiento rectilíneo uniformemente acelerado): Cuando la hoja esté acelerando o desacelerando, seguirá las leyes del movimiento rectilíneo uniformemente acelerado.

- Movimiento parabólico: Cuando la hoja salte, seguirá las leyes del movimiento parabólico, lo que permitirá al jugador controlar la distancia y la altura del salto.

- MCU (Movimiento circular uniforme): La hoja también puede tener una trayectoria circular constante, siguiendo las leyes del movimiento circular uniforme.

- Colisiones: Si la hoja choca con un pájaro, seguirá las leyes de la colisión elástica, lo que significa que la hoja y el pájaro rebotarán en diferentes direcciones con la misma cantidad de energía

**Vista:**

El juego se presentará en tercera persona, lo que permitirá al jugador ver a la hoja de papel desde diferentes ángulos y apreciar mejor el entorno en el que se encuentra. El jugador podrá cambiar la vista frontal para apreciar mejor los detalles y obstáculos que se presenten en su camino.

**Requerimientos**

* Físicas: El juego requerirá la implementación de diversas físicas para proporcionar una experiencia de juego realista y emocionante.
* Sistema de puntuación: el juego requerirá un sistema de puntuación que permita al jugador competir consigo mismo y con otros jugadores.
* Niveles y entornos: el juego necesitará varios niveles y entornos diferentes para mantener la variedad y el interés del jugador.
* Enemigos: los enemigos como los pájaros o las situaciones climáticas se requieren para darle dificultad al juego.
* Jugador: para darle jugabilidad al usuario requerimos poder mover el personaje principal, implementar sus mecánicas y su interacción con los demás objetos del juego.
* Lugares seguros: el objetivo del juego es encontrar un lugar seguro para la estancia de la hoja, por lo que es de vital importancia definir cuáles son y que características tienen los lugares seguros.

**Objetos**

* Game

Atributos

int level: representa el nivel actual en el que se encuentra el jugador.

bool gameOver: indica si el juego ha terminado o no.

Métodos

void startGame(): inicializa el juego y establece los valores iniciales para score y level.

void endGame(): cambia el valor de gameOver a true y muestra la puntuación final del jugador.

void nextLevel(): aumenta el valor de level en 1 y actualiza el entorno y los desafíos para el siguiente nivel

bool checkGameOver(): comprueba si el juego ha terminado, ya sea porque la hoja encontró un hogar seguro o porque fue destruida por los elementos climáticos.

* Player

Atributos:

int character\_x\_size: indica cuantos pixeles de ancho tiene cada sprite de la hoja

int character\_y\_size: indica cuantos pixeles de alto tiene cada sprite de la hoja

int x: posición horizontal del jugador en el mapa.

int y: posición vertical del jugador en el mapa.

float speed: la velocidad de movimiento de la hoja

int health; puntos de vida de la hoja

Métodos:

void move(x, y): mueve la hoja a las coordenadas (x, y) en el mapa

bool is\_dead(): retorna un booleano que permite saber si la hoja sigue con vida

void flotar(): Este método permitirá al jugador controlar la velocidad de descenso de la hoja para evitar obstáculos o prolongar su tiempo en el aire.

void fold(): Permite cambiar la forma de la hoja para adaptarse a los obstáculos y pasar a través de espacios estrechos.

void update\_health(): Método que se encargará de actualizar la vida de la hoja en función del clima y las colisiones con obstáculos.

* Map

Métodos

void generar\_mapa(): Este método se encarga de generar aleatoriamente el mapa para cada nivel.

void actualizar\_mapa(): Este método actualiza la representación visual del mapa de acuerdo a las acciones del jugador.

void colisiones (): Este método se encarga de manejar las colisiones entre el jugador y los elementos del mapa.

* Bird

Atributos

int bird\_x\_size: indica cuantos pixeles de ancho tiene cada sprite del pájaro

int bird\_y\_size: indica cuantos pixeles de alto tiene cada sprite del pájaro

int x: posición horizontal del enemigo (pájaro) en el mapa.

int y: posición vertical del enemigo en el mapa.

float speed: la velocidad de movimiento del enemigo

int behavior: indica el comportamiento actual del enemigo (sentido en el que se mueve)

Métodos

void spawn(): Método que genera un enemigo en una posición aleatoria del mapa.

void move(): Método que determina la dirección y la velocidad del movimiento del enemigo en cada fotograma del juego.

int get\_position(): Método que devuelve la posición actual del enemigo en el mapa.

void set\_position(): Método que establece la posición del enemigo en el mapa en una posición determinada.

* climate\_enemies

Atributos

enemy\_type: el tipo de enemigo climático (por ejemplo, "sol", "lluvia", "tormenta").

int duration: la duración del evento climático en segundos.

int damage: la cantidad de daño que inflige el evento climático al jugador.

bool active: un indicador booleano que indica si el evento climático está activo o no.

int x: posición horizontal del enemigo en el mapa.

int y: posición vertical del enemigo en el mapa.

Métodos

void start(): inicia el evento climático, activando su indicador booleano "active".

void stop(): detiene el evento climático, desactivando su indicador booleano "active".

void pdate(): actualiza la posición del evento climático en el mapa.

int get\_remaining\_time(): devuelve el tiempo restante para que el evento climático termine.

* Puzles

Atributos

string question: La pregunta o enunciado del puzle que el jugador debe resolver.

int answer: La respuesta correcta del puzle.

Métodos

void show\_puzzle(): Muestra la pregunta o enunciado del puzzle al jugador.

void check\_answer(player\_answer): Comprueba si la respuesta proporcionada por el jugador es correcta. Si la respuesta es correcta, el jugador avanzará al siguiente nivel. Si la respuesta es incorrecta, el jugador tendrá que intentar resolver el puzzle de nuevo.

void get\_hint(): Proporciona una pista o ayuda al jugador para resolver el puzzle, si está disponible. La cantidad y la calidad de las pistas dependerá de la dificultad del puzzle

* Safe\_zone

Atributos

int size\_x; tamaño en x de la zona segura

int size\_y; tamaño en y de la zona segura

int pos\_x; posición en x de la zona segura dentro del mapa

int pos\_y; posición en y de la zona segura dentro del mapa

Métodos

void generate\_safe\_zone: método que toma como parámetro el mapa sobre el que se creará la zona segura. Este método generará una zona segura rectangular y la ubicará en el mapa.

bool check\_collision: verifica si Player está colisionando con la zona segura.

* mainwindow

Esta clase forma parte de la librería de Qt y será de utilidad para crear y manipular la interfaz gráfica del juego.