Universidad Panamericana

Materia:

Estructuras de Datos y Algoritmos II

Título:

Proyecto Parcial 2/Aztlan

Nombre del profesor:

Ricardo Tachiquín Gutiérrez

Nombre de los Alumnos:

Aldo Cartamin Pompa/0275866

Emilio Uriel Sánchez López/0275598

Sofía Reyes Salas/0273806

Fecha de entrega:

02/06/2025

UNIVERSIDAD
Panamericana

Aztlan

Autores:
Aldo Cartamin Pompa
Emilio Uriel Sánchez López
Sofía Reyes Salas

Inspirado en el videojuego “Dwarf Fortress” desarrollado por Bay 12 Games, Tarn Adams y Zach Adams.

Sinopsis:

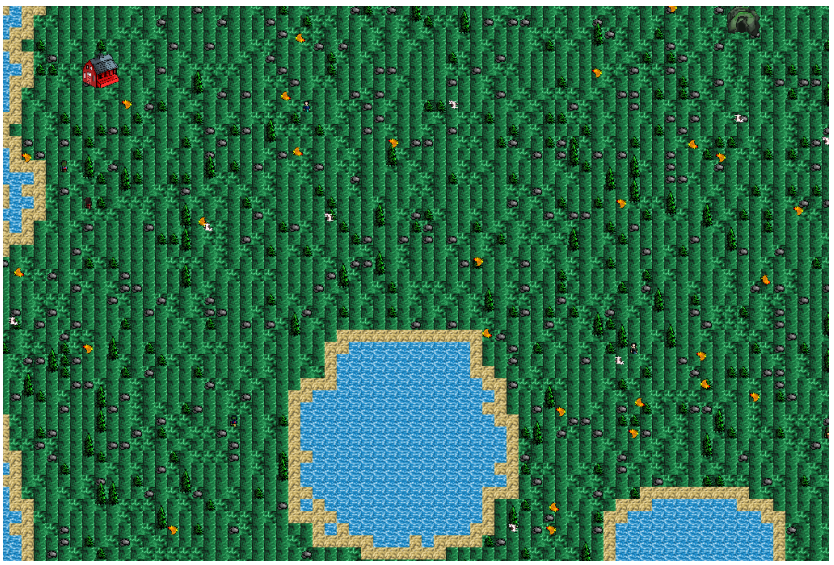
Aztlan es un videojuego de supervivencia ambientado en un mundo post apocalíptico devastado por un virus zombie, la humanidad se encuentra al borde de la extinción debido a que el virus comenzó a expandirse rápidamente por el mundo cobrando miles de vidas y obligando a los sobrevivientes a refugiarse en las zonas rurales alejadas de la ciudad, donde deberán recolectar recursos, construir, cooperar y resistir para dar inicio a la nueva civilización en medio del caos.

Arquitectura:

La arquitectura de Aztlán se desarrolló siguiendo los principios de Programación Orientada a Objetos (POO) aprendidos durante los semestres anteriores. El uso de clases, herencia, encapsulamiento y polimorfismo nos permitieron plantear de una manera más eficiente los objetivos, como las interacciones entre las entidades del mundo, y ayudó enormemente a resolver los distintos obstáculos encontrados a la hora de empezar a darle vida al proyecto.

Arte:

El arte visual de Aztlan fue creado a mano en un 100% por los integrantes del equipo, adoptando un estilo pixel art original y artesanal que le da una personalidad única y le otorga identidad visual al videojuego



Herramientas:

Aztlán fue hecho en el lenguaje de Python utilizando el framework pygame y el arte se desarrolló en el programa de libresprite aprovechando su flexibilidad para la creación de sprites animados y entornos pixelados. Se tomaron como referencia videotutoriales de YouTube, se consultaron libros de programación competitiva y se solicitó la ayuda de inteligencias artificiales para lograr un trabajo más limpio, eficiente y formativo.

A continuación se presenta una descripción más detallada de la estructura interna de Aztlan y la composición de sus clases más importantes, explicando qué función cumple cada una dentro del sistema y el cómo se relacionan entre sí, dando forma al esqueleto del videojuego.

Clases

Ser vivo:

La clase ser vivo representa la entidad biológica base dentro del ecosistema del mundo. A partir de esta se modelan los demás seres con comportamientos autónomos como los humanos, animales y enemigos. Los cuales interactúan entre ellos por medio de sus métodos.

Cada instancia posee atributos físicos, emocionales y fisiológicos que cambian con el tiempo y se ven afectados por sus acciones y necesidades. Esta clase también gestiona el movimiento y los cambios en el estado vital, tales como dormir o morir. Siendo la base para las futuras subclases.

Parámetros de la clase:

- edad (int): Es la edad de cada ser vivo.
- nombre (string): Nombre que diferencia un ser vivo de otro.
- genero (string): Género biológico del ser vivo.
- fuerza (int): Valor que indica la potencia física del ser vivo.
- resistencia (int): Influye en la cantidad de daño que recibe el ser vivo ante un ataque.
- x, y (int): Son las coordenadas de cada ser vivo en el mapa.

Atributos de la clase:

- vida: Estado vital general del ser vivo (0 significa muerte).
- salud: Nivel actual de bienestar físico del ser vivo.
- energia: Nivel de energía del ser vivo.
- hambre, sed: Necesidades básicas que disminuyen con el tiempo.
- felicidad: Estado emocional general del ser vivo.
- velocidaX, velocidadY: Dirección y magnitud del desplazamiento.
- destino: coordenadas hacia las que el ser se está moviendo.
- imagen: Imagen o “sprite” del ser vivo (se define de acuerdo a las clases que heredan de Ser vivo).

Métodos:

comer(): Alimenta al ser vivo, restaurando su nivel de hambre y reduce la cantidad de alimento disponible.

beber(): Hidrata al ser vivo, restaurando su nivel de sed y reduce la cantidad de agua disponible.

dormir():Permite al ser vivo descansar y recuperar energía según el tiempo dormido.

mover(): Gestiona el desplazamiento del ser vivo dentro del entorno.

Si no se especifica un destino explícito, el ser se moverá aleatoriamente dentro de un rango reducido.
morir(): Elimina al ser vivo del entorno.

Las siguientes clases heredan de Ser vivo:

Humano:

Los humanos son la base del juego y de su gameplay, cada humano realiza tareas con el objetivo de asegurar la supervivencia de la comunidad. Cuenta con atributos adicionales a los del ser vivo que distinguen un ser pensante de uno salvaje, los humanos actúan en comunidad, lo que hace que la supervivencia sea más llevadera.

Parámetros de la clase: Se incluyen los parámetros vistos en la clase Ser vivo

- linaje (string): Origen familiar (puede afectar habilidades)
- profesion (string): Ocupación del humano (también puede afectar habilidades)
- estatus (string): Posición social dentro de la comunidad
- bendicion (string): Don o favor divino otorgado (afecta habilidades)
- habilidades(Habilidades): Son las competencias técnicas básicas que todo humano posee. Según cuanto realice una tarea, aumentarán sus habilidades relacionadas.
- tarea (Tareas): Referencia a la tarea actual asignada al humano.
- eficiencia (int): Este parámetro decide quien hace una tarea específica, quien tenga el valor más alto, será el designado para llevarla a cabo.

Métodos: Se incluyen los métodos vistos en la clase Ser vivo.
calcularEficiencia(): calcula la eficiencia de cada ser humano para ver quien hace la tarea designada en el momento.
minar(): El humano busca la roca más cercana y va a recolectarla.
talar(): El humano busca el árbol más cercano y va a talarlo.
construir(): El humano construye un edificio indicado en una posición indicada.
cocinar(): El humano cocina la carne cruda, haciendo que tenga mayor saciedad.
cazar(): El humano busca el animal indicado más cercano y se dispone a cazarlo.
buscar(): Es el método llamado en los anteriores métodos, se le asigna un objeto a buscar y se dispone a explorar el terreno en busca de dicho objeto.

Enemigo:

Los enemigos son los encargados de dar dificultad al juego, estos están esparcidos por el mapa y son agresivos hacia los humanos. La clase enemigos tiene como subclases a Zombie y Bruja, ambas clases tienen los mismos atributos.

Parámetros de la clase: Se incluyen los parámetros de la clase Ser vivo.
Métodos: Se incluyen los métodos vistos en la clase Ser vivo.

Animal:
Los animales son la principal fuente de alimento del juego. La clase Animal tiene como subclases a Vaca y Pollo, ambas clases tienen los mismos atributos.

Parámetros de las clases: Se incluyen los parámetros de la clase Ser vivo

- volumen (int): Valor que indica el espacio que toman al estar dentro de un corral o granero (vaca = 10, pollo = 2)

Métodos: Se incluyen los métodos vistos en la clase Ser vivo.

Aquí terminan las clases que heredan de Ser vivo.

Terreno:
El terreno es de lo que está hecho el mundo, es sobre lo que se mueven los seres vivos del juego y determina si la flora puede crecer sobre él, la clase terreno tiene como subclases a Pasto, Agua y Tierra, todas comparten el mismo atributo.

Atributos:

- imagen: Es la representación gráfica del terreno

La clase Terreno tiene además otras subclases más complejas como Estructuras, de la cual también heredan aún más clases.

Estructuras:
Las estructuras son los objetos visibles posicionados en el mundo y con los cuales puedes interactuar para obtener información o materiales.

Atributos: Se incluye el atributo¿o imagen de la clase Terreno

- nombre: Es el nombre de cada estructura, con la diferencia de que las estructuras del mismo tipo comparten el mismo nombre.

- alto, ancho: Son los valores que determinan el espacio que ocupa la estructura en el mundo.
- fila_base, col_base: son las coordenadas de su ubicación en el mapa.
- botones: Es una lista donde se almacenarán los botones necesarios que cada estructura debe mostrar al hacer click sobre ella.
- mostrar_botones: Es un booleano el cual indica si los botones se están mostrando o no.

Métodos:

crear_botones(): Se encarga de crear los botones necesarios para cada estructura, se define en cada subclase, no en la clase padre.

dibujar_botones(): Una vez que ya se sabe que botones mostrar, este método se encarga de mostrarlos en pantalla.

ocultar_botones(): Una vez que los botones se muestran, cuando ya no se quiere seguirlos observando se manda llamar a este método para ocultarlos.

get_info(): Regresa información necesaria de la clase.

Las siguientes clases se heredan de Estructuras.

Árbol:

Los árboles son fuente de madera en el juego, la cual es necesaria para la construcción de edificios.

Atributos: Se incluyen los atributos de la clase Estructuras

- madera: Es la madera que da a la comunidad cuando se tala el árbol.

Métodos: Se incluyen los métodos de la clase Estructuras

accion_talar(): Cuando un humano intenta talar un árbol se llama a este método para determinar si se puede realizar la acción o no.

eliminar_arbol(): Elimina el árbol del mundo y desocupa al humano encargado.

Roca:

Las rocas son fuente de piedras en el juego, las cuales son necesarias para la construcción de edificios.

Atributos: Se incluyen los atributos de la clase Estructuras

- piedra: Es la piedra que da a la comunidad cuando se rompe la roca-

Métodos: Se incluyen los métodos de la clase Estructuras

accion_romper(): Cuando un humano intenta romper una roca se llama a este método para determinar si se pueda realizar la acción o no.

romper_roca(): Elimina la roca del mundo y desocupa al humano encargado.

Granero:

El granero es el edificio principal de la comunidad, en él se puede ver el inventario de la comunidad.

Atributos: Se incluyen los atributos de la clase Estructuras

Métodos: Se incluyen los métodos de la clase Estructuras.

accion_info(): Imprime la información del granero.

Mina:

En la mina se extraen materiales, útiles para la construcción.

Atributos: Se incluyen los atributos de la clase Estructuras

- roca: es la cantidad de roca que se otorga cuando un humano va a minar.

Métodos: Se incluyen los métodos de la clase Estructuras

accion_minar(): Determina si el humano puede ir o no a la mina y cuantos materiales extrajo de esta.

minar(): Si se determina que el humano sí puede minar, se llama a este método para realizar los movimientos en los inventarios y listas.

Aquí terminan las clases que heredan de Estructuras y de Terreno

Comida:

La comida es el principal alimento del ser humano en el juego, esta se consigue de la caza de animales y se comparte entre todos los miembros de la comunidad

Atributos:

- saciar: Es la cantidad de hambre que se repone al ser consumida.
- cantidad: Es el monto de comida restante en la comunidad.

Agua_bebida:

El agua es la fuente de hidratación dentro del juego, se puede recolectar de alguna fuente de terreno de agua, como ríos o lagos, se comparte entre todos los miembros de la comunidad.

Atributos:

- hidratar: Es la cantidad de sed que se repone al ser consumida.
- cantidad: Es el monto de agua restante en la comunidad.

Comunidad:

La comunidad es donde los humanos toman decisiones y administran los recursos del mundo, funge como el intermediario entre el jugador y las decisiones que toman los aldeanos.

Atributos:

- inventario: Es una instancia de otra clase llamada Inventario, la cual solamente cuenta con variables como: agua, comida, madera, roca, etc. Es solo un soporte para mantener el orden y accesibilidad.
- bajoAtaque: Es un booleano que ayuda a reorganizar la prioridad de las tareas pendientes en caso de asedio a la comunidad.
- colonia: Es una instancia de la clase Grupo_humanos la cual solo tiene ese nombre pero su comportamiento es el de una DoublyLinkedList, esta nos ayuda a mantener organizados a los humanos.
- ocupados: Es otra instancia de Grupo_humanos esta nos ayuda a saber a que humanos no podemos asignar tareas ya que ya se encuentran realizando una.
- sin_ocupar: Nuevamente otra instancia de Grupo_humanos, con la diferencia de que esta es una copia directa de colonia, pues al empezar el juego todos los humanos están desocupados, con ella sabemos a qué humanos si podemos asignar tareas.
- tareasPendientes: Es una instancia de TareasQueue, la cual actúa como una queue normal sin nada extra, aquí se almacenan las tareas antes de ser ordenadas por el heap.
- tareasEnProgreso: Son las tareas que ya fueron ordenadas y asignadas a un humano pero que todavía no han sido finalizadas.
- heap: Es una instancia de TareasPriorityQueue, aquí llegan las tareas que salen de tareasPendientes, se ordenan según su prioridad en ese momento y salen para ser asignadas a un humano que posteriormente las llevará a cabo.

Métodos:

humanoMasApto(): Busca al humano más apto para cierta tarea que se encuentre desocupado y lo regresa.

asignarTareas(): Toma una cantidad de tareas pendientes que coincida con la cantidad de humanos desocupados que hay en ese momento, saca las tareas de la queue y las manda para el heap, una vez ordenadas, comienza a llamas a humanoMasApto, para que a cada tarea se le asigne el mejor candidato.

avanceTareas(): Si una tarea es demasiado larga, esta función se encarga de que con el pasar de los días, el progreso de la tarea se vea reflejado.

A continuación se dará una pequeña demostración de cómo es la lógica detrás de las tareas, los grupos de humanos, el heap, etc.

Cuando creamos una comunidad comenzamos con 6 humanos los cuales se pueden categorizar de la siguiente forma:

Grupo_humanos (DoublyLinkedList)

colonia	Uriel <-> Aldo <-> Sofia <-> Daniel <-> Karol <-> Adalberto
ocupados	Sin ocupados
sin_ocupar	Uriel <-> Aldo <-> Sofia <-> Daniel <-> Karol <-> Adalberto

Tareas (Queue, LinkedList y PQueue)

tareasPendientes	recolectar madera <- construir_edificio <- recolectar_agua
tareasEnProgreso	Ninguna tarea en progreso
heap	Sin tareas

Ahora queremos que los humanos se pongan a realizar tareas, por lo que sacamos las tareas de la queue y las mandamos al heap.
El heap las ordena y quedan de la siguiente manera:

tareasPendientes	No hay tareas pendientes
tareasEnProgreso	Ninguna tarea en progreso
heap	recolectar_agua <- recolectar_madera <- construir_edificio

Luego se toma al humano más apto para cada tarea del grupo sin_ocupar, se le asigna la tarea y se mueve al grupo ocupados, al mismo tiempo comienza a ejecutarse la tarea.

colonia	Uriel <-> Aldo <-> Sofia <-> Daniel <-> Karol <-> Adalberto
ocupados	Uriel <-> Aldo <-> Sofia
sin_ocupar	Daniel <-> Karol <-> Adalberto

tareasPendientes	No hay tareas pendientes
tareasEnProgreso	recolectar_agua <> recolectar_madera <> construir_edificio
heap	Sin tareas

Y finalmente cuando se terminan de realizar las tareas vuelve todo a como estaba.

colonia	Uriel <-> Aldo <-> Sofia <-> Daniel <-> Karol <-> Adalberto
ocupados	Sin ocupados
sin_ocupar	Uriel <-> Aldo <-> Sofia <-> Daniel <-> Karol <-> Adalberto

tareasPendientes	Sin tareas pendientes
tareasEnProgreso	Ninguna tarea en progreso
heap	Sin tareas

Así se concluye el ciclo de tareas de la comunidad.

Ideas para futuras implementaciones:

- Añadir sistema de conexiones y lazos entre la comunidad.
- Aumentar los eventos al azar.
- Mejorar las interfaces.
- Implementar mejores estrategias para animaciones.