Universidad Nacional Autónoma de México

Proyecto 2: Call of Chienchias

Modelado y programación

Equipo:
NoSeQueHacerConMiVidaException

Integrantes:

- Martínez Cano Ricardo Iván 320283284
- Méndez Razo Carlos Geovanni 315057249
- Vidal Aguilar Diego Jesus 319297591

1. Problematica: La alfa de un juego

Al equipo NoSeQueHacerConMiVidaException nos ha llegado nuestro jefe con una petición, el mismo buscaba que realizaramos el alfa de un juego, el cual podamos presentar frente a la junta directiva y esperar que la idea pase a considerar como un juego al cual invertir una cantidad considerable de dinero para su desarrollo y distribucion a futuro.

Como ya se imaginarán, el jefe nos ha presionado para que su equipo sea el que se lleve el dinero para el desarrollo del juego, asi mismo no conforme con buscar que seamos los ganadores, quiere que lo hagamos con un tema relativamente reciente bajo el cual captemos la atencion de las personas. Logicamente lo unico en nuestras ideas fue un juego de Chinches, y de esta forma ha nacido Call of Chienchias

Call of Chienchias es presentado como respuesta a la busqueda de la alfa de un juego que nos permita conseguir el interés de la junta directiva para su desarrollo.

Call of Chienchias es un juego pensado en la reciente polémica de las chinches en la UNAM, donde tu como jugador, estarás encargado de lidiar con este desastre exterminando a las chinches que invaden la Facultad.

Esta versión posee un estilo Metroidvania, con un desarrollo visual y mecánico simple debido al corto tiempo y los limitados recursos proporcionados por la empresa para el desarrollo de esta alfa, Call of Chienchias esta pensado para que te puedas sumergir en la vida de un estudiante de ciencias, en particular de un Computólogo, pensando en como este acabaría con las chinches que han invadido su facultad.

Encontrándos sobre el camino distintos aliados que te permitirán recibir las mejoras adecuadas para tu aventura, podrás interactuar con distintos profesores, recibir la ayuda de alguno de tus amigos y ¿por qué no? hasta ser afectado por alguno de ellos. El alfa de Call of Chienchias se encuentra ambientado en nada mas y nada menos que la facultad de ciencias.

Liberando a ciencias de las chinches, te encontraras con distintos enemigos que no te lo dejaran tan sencillo, mientras que para esta alfa nos dimos a la tarea de solo implementar 4 enemigos, estos 4 enemigos contaran con propiedades distintas, incluso algunos con diseños unicos que te dejaran asombrado.

En el diseño de esta alfa nos podemos encontrar con distintos detalles buscando que el juego pueda ampliarse a futuro de una manera mas sencilla, si bien se busca respetar la idea inicial que hemos construido a lo largo de nuestro proyecto, se ha pensado en un proyecto que permita la adicion de nuevos NPC, nuevos aliados que te permitan cambiar el rumbo de una pelea en un instante, nuevas armas y junto con ellas nuevos personajes jugables, si bien la implementacion fue pensada en un computologo, se ha dejado el codigo con la estructura que permitira su correcta modificacion para posteriormente poder generar un Actuario, Un biologo, Un fisico y por que no, incluso los ingenieros son bienvenidos.

Call of Chienchias no fue pensado para que su unico mapa fuera ciencias, posteriormente se podran ir agregando mapas de acuerdo a un novedoso sistema que te permite generar colisiones automaticamente a partir de una base de instrucciones y un mundo nuevo.

2. Patrones utilizados

- Strategy: El patrón strategy nos permite crear distintos tipos de ayudantes con imagenes distintas y ataques distintos, donde todos surgen a partir de una interfaz *Aliados* de donde saldrán las estrategias concretas que serán los estudiantes de las clases *Biólogo, Físico*, etc. Mientras que nuestra clase contexto será la clase *Teléfono* ya que almacenará las referencias de estas estrategias concretas para posteriormente ser usadas por el jugador. El patron Strategy tambien es usado en Rutas, esto con el fin de poder correr el juego en distintos entornos, para ello se hacen uso de metodos que cambien la ruta de acuerdo con el entorno en el que se esta trabajando

- Prototype: El patrón prototype nos permite crear varias copias de instancias objetos ya existentes, por lo que será utilizado para la generación de multiples enemigos provenientes de una misma clase de chinche el cual el jugador deberá derrotar. En nuestro caso la interfaz prototipo será la clase abstracta *Enemy* ya que es donde se declara el método de clonación, mientras que los prototipos concretos seran las clases *ChincheChikita, ChincheGrandota, Chinchientifica* y *ChincheDirectora* serán los prototipos concretos que implementarán este método de clonación.

- Decorator: Este patrón permite modificar propiedades de objetos al encapsularlos dentro de otros objetos, en particular, lo utilizamos para alterar los valores del jugador a través de la clase *AtributosPlayer*. Esta será la clase decorada donde se modificaran valores como velocidad, daño de ataque y cadencia del arma cuando interactua con los NPC. En este caso las clases decoradoras serán las que implementan la interfaz *Arma* tales como *Java, Python, C, Cplusplus* y *PHP* ya que representan las armas que va adquiriendo un computólogo al momento de avanzar en su carrera e interactuar con distintos profesores que le van enseñando nuevos lenguajes de programación.

- State: Nos permite movernos entre distintas fases del juego, tales como el menú, el propio juego, la pausa, pantalla de muerte y de victoria. Debido a problemas con los paneles, este patrón no esta implementado apropiadamente, por lo que al ejecutar no se vera el menu todavia.

- Facade Este patron nos permite que al iniciar el juego se inicien varios procesos como crear los niveles, manejar el movimiento del jugador, o reproducir la musica, etc.

- MVC: Las componentes del modelo MVC se implementan de la siguiente forma; el Modelo estará compuesto por aquellas clases que contengan los datos del juego, como las clases que albergan entidades e items (*Entity, Item, Player,NPC, Aliados, Enemy*) y sus respectivas subclases. También aquellas clases que permiten recuperar las imagenes y sonidos del juego como *AssetSetter, SpriteSheet, SoundPlayer*. También el modelo incluye la lógica entre estos datos, como la clase *CollisionChecker*. El controlador estará dado por la clase *GamePanel* ya que es la que reacciona directamente a las interacciones del usuario, enviando peticiones o updates a las entidades para

que inmediatamente se muestren en la Vista, que también es controlada por esta clase ya que es encargada de mostrar la interfaz gráfica que visualiza el usuario.

3. notas

- Colisiones

  Las colisiones se trabajaron lo mejor que pudimos y de la manera mas optima que pudimos, considerando que se trata de la alfa de un juego es normal que presente una serie de errores en cuanto a las colisiones, todos lo tienen, y el nuestro no es la excepcion, el retroceso de los enemigos puede llegar a sacarte del mapa debido a que atraviesa las colisiones

  Se trabajaron de la manera mas optima posible y se busco hacerlo lo mas limpio posible, haciendo que en muchas ocasiones el objeto sea el que se pregunte si esta colisionando, en lugar de preguntarse si ya colisiono con algo.

- Cooldown

  Para muchos de nuestros controles y accion se trabajo bajo cooldown, esto porque la manera en la que leemos los controles es constante, entonces al leer la pulsacion de una tecla realmente estas leyendo las cientos de veces que se esta pulsando la misma tecla en ese corto periodo de tiempo. Por ello se opto en poner en algunos casos un cooldown de medio segundo que nos permitiera no registrar mas de una accion por medio segundo

- La optimizacion

  EL juego se ha intentado optimizar de la mayor manera posible, pero sigue tirando mucho del procesador y el hecho de que sea un juego diseñado en Java no ayuda demasiado

- La pantalla muerte y win

  Ambas pantallas y estados consecuentes son unicamente pantallas en las cuales el acceso del usuario esta limitado a observar un mensaje, es decir, solo aparece una pantalla con la notificacion de morir o vivir, incluso sin la opcion de mas actionKey, pensando en que el juego no pueda ser reiniciado si no es desde el comienzo todo

- Keyboard

  Ocurrio un problema con el teclado entre los distintos estados, el teclado no es detectado correctamente en el cambio de estados, por ello se opto por ponerlo en recursos como una clase con un unico metodo estatico, pero esto nos sigo acarreando complicaciones, la principal de ellas, es que no lo lee correctamente Debido a que no sabemos por que ocurre esto, solo podemos presentar por el momento una debida solucion, y es que el teclado solo funciona en el cambio de estados si se preciona con el raton fuera de la pantalla del juego y se vuelve a presionar dentro de la pantalla del juego, siendo este un problema a resolver en las futuras actualizaciones

- Opcion instrucciones y creditos

  Estas opciones se quedaron como meramente opciones visuales para indicar lo que proximamente podria agregarse o deberia agregarse a la alfa del juego, siendo mas opciones de relleno que recomendamos encarecidamente no presionar (El juego se va a un tremendo bucle si lo hace) que posteriormente deberían ser implementadas

- Controles:

  $\rightarrow$ mover hacia la derecha
  $\leftarrow$ mover hacia la izquierda
  $\uparrow$ mover hacia arriba

$\downarrow$ mover hacia abajo
a ataque a la izquierda
b ataque a la derecha
w subir escaleras
s bajar escales
$\{1, 2, 3, 4, 5\}$ invocar aliados
q ocupa potenciadores adquiridos de los NPC

4. Referencias

- RyiSnow (2021, November 14). How to make a 2D game in Java [Video]. Recuperado de `https://youtu.be/om59cwR7psI?si=kcAHvKrPtqT7W0R6`
- KUWAGO (2023, March). Swinging. [Song] Toy Box #5. Recuperado de `https://soundcloud.com/kuwago-1`
- Sprites jugador principal: "Chicken Boy Pack" (2021), del usuario @9E0_D0. Recuperado de `https://9e0.itch.io/chick-boy`.