**UNIVERSIDAD ICESI**

*Facultad de Ingeniería*

*Algoritmos y Estructuras de Datos*

*Periodo II de 2018*

**Laboratorio dos Fortnite**

C. Gironza Mamian1,C. Olano Rosas2, N. Salazar1,

*1. Ingeniería telemática, Facultad de ingeniería, Universidad Icesi, Lina Marcela Quintero Villareal, Cali Colombia*

2*Ingeniería de Sistemas, Facultad de ingeniería, Universidad Icesi,* *Norha Milena Villegas Machado, Cali Colombia*

Introducción

En el siguiente informe se registrará todos los pasos del proceso de ingeniería que se llevaran a cabo para la solución del laboratorio dos de la segunda unidad del curso de Algoritmos y Estructuras de Datos (AED) de la universidad ICESI; este laboratorio tiene como temática el juego online Fortnite.

Contexto problemático:

Las preocupaciones de la compañía Epic Games son claros el primero es: la dificultad a la cual se enfrenta un usuario para encontrar una partida en la cual todos los participantes estén a su mismo nivel, los síntomas de esta preocupación son:

* El ping de cada jugador
* La posición geográfica de los jugadores de forma individual

La segunda preocupación de la empresa está relacionada con crear partidas que sean en modo plataforma, los síntomas de esta exigencia podrían ser:

* No todos los jugadores de Fornite juegan en las mismas plataformas, (Pc, Xbox Play Station) con esta modalidad de juego podría llegar a ser más fácil el manejo de los servidores

La última recomendación del equipo de Epic Games es la implementación de una nueva modalidad de juego, con la temática principal del día de San Valentín, ¿razón de esta última propuesta?, la necesidad de seguir innovando el juego con diferentes temáticas.

1. Identificacion del problema

El juego Fornite ha recibido varias recomendaciones de sus usuarios, estos aseguran que necesitan que los servidores agrupen por niveles a los usuarios y así no generar partidas aburridoras, además de que se les agrupe dependiendo a la plataforma del usuario. Por otro lado, al ser un juego en línea, necesita ir innovando es sus modalidades y así asegurar su permanencia en el mercado.

* 1. Requerimientos

|  |  |
| --- | --- |
| 1. **Nombre** | **R1. Clasificar jugadores por plataforma** |
| **Resumen** | Clasifica a los jugadores según la plataforma a la cual pertenecen |
| **Entradas** | |
| Plataforma en la cual el jugador accede al juego | |
| **Resultados** | |
| El jugador ha sido asignado al grupo que le corresponde según su plataforma | |

|  |  |
| --- | --- |
| **Nombre** | **R2. Clasificar jugadores por ping** |
| **Resumen** | Asigna a un jugador en un grupo determinado el cual corresponda con el intervalo al cual pertenece su ping |
| **Entradas** | |
| Ping del jugador | |
| **Resultados** | |
| El jugador ha sido asignado al grupo que le corresponde según su ping | |

|  |  |
| --- | --- |
| **Nombre** | **R3. Clasificar jugadores por experiencia en el juego** |
| **Resumen** | Clasifica a los jugadores según su experiencia en el juego (esto basado en diferentes criterios) |
| **Entradas** | |
|  | |
| **Resultados** | |
| El jugador ha sido asignado al grupo que le corresponde según su experiencia | |

|  |  |
| --- | --- |
| **Nombre** | **R4. Determinar nivel del jugador** |
| **Resumen** | Determina el nivel del jugador con sus características |
| **Entradas** | |
|  | |
| **Resultados** | |
| Se ha clasificado al jugador | |

|  |  |
| --- | --- |
| **Nombre** | **R5. Clasificar jugadores por plataforma** |
| **Resumen** | Clasifica a los jugadores según la plataforma a la cual pertenecen |
| **Entradas** | |
| Plataforma de juego | |
| **Resultados** | |
| El jugador a sido asignado al grupo que le corresponde según su plataforma | |

|  |  |
| --- | --- |
| **Nombre** | **R6. Determinar geolocalización** |
| **Resumen** | Determina en qué lugar en cual se encuentra el jugador |
| **Entradas** | |
|  | |
| **Resultados** | |
|  | |

|  |  |
| --- | --- |
| **Nombre** | **R7. Eliminar arma** |
| **Resumen** | Elimina el arma que cuya cantidad de armas sea igual a cero |
| **Entradas** | |
|  | |
| **Resultados** | |
| El arma ha sido eliminada | |

|  |  |
| --- | --- |
| **Nombre** | **R8.** |
| **Resumen** |  |
| **Entradas** | |
|  | |
| **Resultados** | |
|  | |

1. Recopilación de la información

Para la solución de este problema hemos tenido que investigar que es Fortnite, cuáles son sus modalidades de juego; además de investigar que es el ping, que es la latencia, como funciona la geolocalización.

1. ¿Qué es Fortnite?: es un juego que combina las modalidades de juego de construcción y de disparo en un solo hibrido.
2. ¿cueles son sus modalidades de juego?: las modalidades de juego que los usuarios pueden escoger son:

* Fortnite Battle Royale, consiste en la batalla de un grupo de personas que se conectan en línea hasta que una sola persona “queda en pie” en este modo de juego se van creando zonas seguras las cuales encierran a los jugadores; hasta esta parte el juego no parece nada del otro mundo, cuya temática es igual a la de muchos otros juegos, pues bien la diferencia de este juego con otro es que los jugadores pueden crear estructuras con diferentes materiales que pueden recolectar del mundo en el cual se encuentran.
* Fortnite: Salvar el Mundo, consiste en colocar 4 jugadores en un escenario postapocalíptico, en el cual cooperan para logran una serie de objetivos en común mientras intentan sobrevivir de oleadas de zombis.

1. ¿Qué es latencia?: es el tiempo necesario para cargar la conexión con un servidor, es decir, decimos que una conexión tiene una alta latencia (baja velocidad para conectarse a la red), cuando el ancho de banda de esta conexión es bajo, y de forma reciproca diremos lo contrario cuando el ancho de banda sea más alta.
2. ¿Qué es el ping?: también conocido como Packet Internet Groper es una herramienta usada para diagnosticar el estado de conexión de una red, esto se hace ejecutando este comando en la consola (también llamada cmd o terminal, según algunos sistemas operativos)
3. ¿Qué es geolocalización? La geolocalización es un proceso a través del cual se puede obtener las coordenadas de una persona o dispositivo (esto gracias los diferentes tipos de geolocalización como GPS, GSM y WIFI)
4. Búsqueda de soluciones creativas

El método utilizado para la generación de ideas ha sido el de la lluvia de ideas, en la cual todos los involucrados dieron sus ideas las cuales fueron agregadas a la lista sin ser pensadas dos veces.

Ideas creativas:

-Darle al jugador un bonus de nivel (aumentar su nivel), en base a cuanto tiempo es capaz de durar en la partida solo con el arma base (el pico)

-Aumentar el nivel del jugador en base a cuantas personas es capaz de matar solamente con el pico

-Para el nivel de los jugadores solo tener en cuenta el tiempo en el que el mismo está peleando con otros usuarios (esto sería tomado como tiempo en acción)

-Sumar el tiempo recorrido por el jugador a la experiencia de este

-Tener en cuanta el desarrollo del jugador en relación con su ping

-Aumentar el nivel de experiencia del jugador por cantidad de estructuras construidas en la partida

-Para el nivel del jugador tener en cuenta la cantidad de veces que se ha curado el jugador en relación con el tiempo de juego

-Contar la cantidad de disparos recibidos y acertados para disminuir o aumentar el nivel

-Aumentar la experiencia del jugador en base a la relación de tiros acertados y distancia de disparo

1. Diseños preliminares
2. Evaluación y selección de la solución
3. Listado de Información Y especificaciones
   1. TAD

|  |
| --- |
| TAD Partida |
| Servidor = {Jugadores= <>} |
| Invariante  -     ∀ Jugadores : (el ping, la geolocalización y nivel ) ∈ el mismo intervalo |
| Operaciones Primitivas:   * put (K key , V value ):  void * isEmpty() :  Boolean * remove(K key ,V value):  void * replace(V value,V value ):  void |

|  |
| --- |
| Put(K key, V Value) |
| “Agregar un nuevo objeto”  {Pre: no especificada}  {post: un nuevo objeto se a agregado al hash table} |

|  |
| --- |
| isEmpty() |
| “Valida si el hash table está vacía”    {Pre: true}    {post: retorna true si está vacía o false si tiene al menos un objeto} |

|  |
| --- |
| Remove (K key, V value) |
| “borra un objeto de la table hash”  {Pre : key != null , key != “ ” value != null , value != “ ”}  {pos : el objeto ha sido eliminado } |

|  |
| --- |
| Remplace(V value1, V value2) |
| “remplaza un objeto de la tabla hash por otro”  {Pre : value1 != null, value2 != null }  {pos : los valores han sido intercambiados (sus llaves han sido)} |

|  |
| --- |
| TAD ISimpleNodeC |
| ISimpleNodeC |
| Operaciones Primitivas:   * getItem:  void * setItem(T item):  ISimpleNodeC<T> * getNext(): <T> * setNext(ISimpleNodeC<T> node):  void |

|  |
| --- |
| TAD SimpleNodeC |
| ISimpleNodeC = {ítem,next} |
| Operaciones Primitivas:   * getItem:  T * setItem(T item):  void * getNext(): ISimpleNodeC<T> * setNext(ISimpleNodeC<T> node):  void |

|  |
| --- |
| getItem() |
| “Retorna un elemento”  {Pre: elemento!=null}  {post: se ha obtenido el elemento} |

|  |
| --- |
| setItem(T item) |
| “Reemplaza un elemento con el elemento especifiado”  {Pre: item!=null}  {post: se ha reemplazado el elemento} |

|  |
| --- |
| getNext() |
| “Retorna el siguiente elemento”  {Pre: siguiente elemento!=null}  {post: se ha retornado el siguiente elemento} |

|  |
| --- |
| TAD Arma |
| Arma = {nombre, balas} |
| Invariante  -    ∀ arma cuando la munición se agote el arma dejara de existir |
| Operaciones Primitivas:  - crearArma :  - disparar arma actual : → Arma  - lanzarArma : → Arma  - recogerArma : → Arma |

|  |
| --- |
| CrearArma |
| “crea un arma nueva y la agrega a la pila de armas ”  {Pre : no especificada }  {pos : el arma ha sido agregada a la pila } |

|  |
| --- |
| DispararArmaActual |
| “dispara el arma que el judor posee actualmente ”  {Pre : balas != 0 }  {pos : la cantidad de balas ha disminuido} |

|  |
| --- |
| LanzarArma |
| “lanza un arma cuando su cantidad de balas es igual a cero ”  {Pre : balas == 0 }  {pos : el arma ha sido eliminada de la pila} |

|  |
| --- |
| RecojerArma |
| “recoje un arma aleatoria y la agrega a la pila ”  {Pre : no especificada }  {pos : el arma ha sido agregada a la pila} |

|  |
| --- |
| TAD IStackC |
| IStackC |
| Operaciones Primitivas:   * empty():  boolean * peek():  T * pop(): T * push(T item):  void |

|  |
| --- |
| TAD StackC |
| StackC = [first] |
| Operaciones Primitivas:   * empty():  boolean * peek():  T * pop(): T * push(T item):  void |

|  |
| --- |
| Peek() |
| “Muestra el ultimo dato introducido en la pila”  {Pre: empty()==false}  {post: se ha retornado el ultimo elemento introducido} |

|  |
| --- |
| Empty() |
| “Comprueba si la pila esta vacía”  {Pre:}  {post: estado de la pila} |

|  |
| --- |
| pop() |
| “Extrae el ultimo elemento introducido”  {Pre: empty()==false}  {post: se ha extraido el ultimo elemento introducido} |

|  |
| --- |
| push(T item) |
| “Introduce el dato especificado a la pila”  {Pre: }  {post: se ha introducido un nuevo elemento } |

* 1. Pruebas

