**Diapositiva 1**

Buenas, soy Luis Pastor. Y voy a hablaros del Chess Battle Royale. Imagino que por el nombre ya suponéis de que se trata. Chess de ajedrez (el juego estará enteramente inspirado y basado en el ajedrez) y Battle Royale porque será una pelea de todos contra todos (en este caso serán 4 jugadores) donde el terreno de batalla se va reduciendo para que los jugadores tengan que pelearse. Sencillo ¿no?

Bueno así dicho la verdad es que sí, parece sencillo. Dos transparencias más y para casa…

Pero bueno, quizás contar como se ha diseñado, lo que se ha hecho para codificar este juego, tener presente las distintas expansiones que habría que realizar y como está el proyecto sea más complicado.

**Diapositiva 2**

De hecho como señala este índice voy a hablaros de distintos temas.

Primero como siempre hay una introducción donde se intentará explicar cuales son los puntos que se han intentado explicar en esta presentación. Luego intentaremos explicar que creemos que necesitaríamos para desarrollar este proyecto desde un punto de vista empresarial. Hablaremos de que metodología se ha intentado seguir para la realización del proyecto y cual es su estado actual. Pasaremos a introducir los factores que nos han parecido más relevantes durante la codificación y el diseño y pasaremos a leer las conclusiones.

Hasta aquí todo parece una presentación tipo… Y eso hablando de un videojuego suena un tanto …. Digamos que aburrido.

**Diapositiva 3**

Cuando estaba organizando estas diapositivas la verdad es que enseguida noté el problema de que se podría organizar la presentación desde diversos puntos de vista. Si fuera la presentación desde un punto de vista empresarial buscando una posible financiación o captación de socios o inversores es muy distinto de si lo que se trata es de mostrar desde un punto de vista formativo o educacional de lo que se ha hecho y como se han usado las distintas herramientas o si simplemente nos tratamos de centrar en como ha quedado el videojuego creado y que hay que “venderlo”. En un caso nos centrariamos en las posibilidades empresariales, en otro en la metodología y el código y la forma de usarlo, y en el tercero en el marketing para crear hype y compradores.

He intentado entremezclar estos tres puntos de vista durante la presentación así que seguramente la presentación adolecerá de problemas bajo cualquiera de ellos pero es que además este proyecto…. No está finalizado.

Y eso quiero que quede claro desde el minuto 1 de esta presentación. El proyecto se encuentra en construcción, así que lo que voy a contar son las bases de lo que se ha hecho hasta ahora, una especie de foto fija. Y que lo que habría que hacer es ir viendo como se desarrolla este proyecto.

Para ello pueden visitar el repositorio github donde está todo el material que se ha ido haciendo o utilizando. Pinchemos en él.

Como veis es un repositorio github normal… y en el fichero README.md esta toda la descripción de su contenido…

La documentación entregada, las pagina github pages creada junto con los javadoc, los diagramas uml usados para la documentacio, los ejecutables, el proyecto en sí, etc…

Cada uno de estos apartados se pueden visitar, bajar, etc. Y bueno aunque lleva parados basicamente desde la entrega pues ahí es donde se irán subiendo las modificaciones y mejoras al proyecto en sí. Falta una tarea de organizar y limpiar pero bueno recordar lo de las obras.

Si vemos la pagina github pages tiene un enlace a un pequeño tutorial donde se cuentan las modificaciones del Chess Battle Royale frente al ajedrez tradicional, la documentación de las clases en JavaDoc (habría que actualizarlo, pero ya iremos haciéndolo) y te puedes descargar directamente el manual o el ejecutable si no te apañas haciendo un clone del repositorio o buscando en la estructura del readme la carpeta correspondiente.

Hay muchas cosas más que se podría hablar del repositorio (y algunas las veremos más adelante en la presentación) pero bueno no podía seguir sin presentarlo y que se viese en parte el gran trabajo que hay que realizar … Aunque la idea en sí del proyecto parezca sencilla.

Como he dicho antes el proyecto se puede ver desde distintos puntos de vista… comencemos con el punto de vista empresarial.

**Diapositiva 4**

Para emprender una actividad empresarial se necesitan varias cosas aparte de aptitudes y actitudes.

Está claro que se necesita una idea. En este caso está clara cual es la idea ¿no? Pero no se trata de que sea una especie de inspiración sino hay que evaluar esta idea emprendedora, si es viable, si es capaz de producir un beneficio, como se diferencia de los competidores, en que innova y podemos usar como ventaja competitiva.

Quizás está feo que yo lo diga pero aunque hay distintos competidores tanto en el mundo del ajedrez puro (en la documentación pongo ejemplos de todas las herramientas informáticas que rodean al mundo ajedrecístico) bastante bien asentados y productivos creo que el concepto de tablero menguante y algunos otros lo hacen innovador y puede tener parte de “mística” del deporte mental de “ajedrez” con un juego más “casual”. De todas maneras como he dicho puede ser feo que la evalue yo así que dejaré que eso lo hagáis los aquí presente.

Por otra parte se necesita un capital inicial. No he hecho las estimaciones correspondientes (y mira que creo que es lo principal) porque creo que hasta tener un producto base más “vendible” el capital es el tiempo y el esfuerzo que depositemos para hacer eso (no se necesitará un ordenador de la NASA para programar estas cosas). Aún así creo que se podría comenzar desde unos bajos capitales y riesgos iniciales e ir escalando conforme las necesidades empresariales fueran surgiendo.

En cualquier caso el último punto es de una vital importancia. Muchas veces cuando se habla de las necesidades uno se centra, en la idea, en la financiación, en el… y se suele olvidar un capital importantísimo como es el trabajo humano.

Chess Battle Royale no puede desarrollarse solo. Necesita más cabezas pensantes y manos. Y no sólo para programarlo sino para diseñarlo, para venderlo, para publicitarlo, para organizarlo, para…. Sólo es imposible y por eso es necesario la formación de un equipo, la formación de una sociedad.

Creo que el mejor camino a medio y largo plazo sería la formación de una S.L. (posiblemente formación sucesiva) pero (y es un pero grande) preferiría que los socios que la formásemos no solamente tuviésemos la responsabilidad de la formación sino que formásemos parte de la directiva y trabajásemos en el proyecto.

Lo sé, es una cosa demasiado utópica e idealista pero dado lo verde que se encuentra el proyecto creo que es mejor fraguar las bases con socios que trabajasen en el proyecto aunque cueste más encontrarlos y luego buscar más financiación si es necesaria (tampoco se necesita un capital inicial muy alto) que buscar la financiación para contratar gente a posteriori.

Y eso nos lleva al siguiente punto…

**Diapositiva 5**

… la captación de socios…

Y esto es un poco como siempre ocurre en la vida… por una parte está lo que ofrece un proyecto como Chess Battle Royale a quien quiera unirse… Ya sabéis lo de proyecto innovador dentro de una industria creciente como la de los videojuegos y que estando desde cero puedes crecer con nosotros y…

Y bueno luego está la realidad. Que me he permitido ejemplificar con la típica viñeta de Axterix y Obélix…. Alistate a la legión… veréis mundo, buena paga… A mi me gusta ser realista (algunos lo llaman pesimista) y claro: se ofrece mucho trabajo y desvelo por que no se parte de algo ya establecido en un estudio claramente formado y con una presencia ya en el mercado. Aquí hay que hacerlo todo y todo cuesta. Y no hay ninguna seguridad de recompensa. Lo más posible es que acabemos como el soldado romano… Pero bueno, si que puedo prometer que intentaremos pasarlo bien durante la ejecución del proyecto.

Pero bueno antes de alistaros. Imagino que queréis saber como se encuentra el proyecto y que metodología de trabajo se quiere emplear y se está usando… ¿no? Vamos a ver si las siguientes transparencias van aclarando vuestras dudas.

**Diapositiva 6**

Tradicionalmente un ciclo de vida del software era un ciclo en cascada. Se hacía un análisis de los requisitos, conforme a ese análisis se hacía un diseño, luego se codificaba y se realizaban las pruebas. ¿Pero que ocurre cuando el análisis no puede ser un análisis completo o se produce un error? En un ciclo esto…

Hoy en día hay numerosas metodologías para mejorar estos problemas (como p.e. distintias metodologías ágiles). En el Chess Battle Royale he intentado realizar un ciclo de vida donde hubiera más de una iteración o de estos ciclos en cascada (modelo iterativo) cada uno de ellos dando una versión diferente cada vez más cercana al final) pero que tuvieran una funcionalidad al menos en alguna e ir añadiendo funcionalidades (lo que seria un modelo incremental).

Para ver un poco las diferencias entre estos dos métodos pueden ver la viñeta… el método iterativo cada vez es más de un coche pero no se puede usar hasta llegar al final. Y en el otro en cambio se van añadiendo además de versiones funciones…

**Diapositiva 7**

¿Y ahora en que paso nos encontramos? Pues en los primeros ciclos hemos ido introduciendo toda la Lógica ajedrecística (los movimientos, tableros, casillas, etc), tenemos un esbozo de una IA (y posibilidad de añadir más) y hemos hecho que se vea en pantalla como algo más que toString gracias a libGDX. Queda aún cosas ahí por modificar y enlucir (por eso digo que es un work in progress) y esperamos que lo que venga siguiente son las versiones donde se van a ir añadiendo funciones de bases de datos (para eso requeriremos distitnas versiones donde se enlacen más fuertemente la notación, datos de los jugadores como los de los login y los rankings) . Y menos coming soon será el con todo ello establecer por fin una versión con características multiplayer.

He creído que era mejor tener primero esa parte lógica y luego la visual y luego lo que queda porque creo que era el core de la aplicación y que aunque haya que hacer rediseños de esa parte por los requisitos de las demás era más sencillo dejar una estructura adecuada desde cero de esas partes que empecinarse en que funcionase todo a la vez.

Eso me ha llevado a bastante rediseños (p.e. hay parte de la IA que al ponerlo en gráfica aún está rascando) pero bueno siempre se parte en esos casos de una base y no desde un análisis desde cero que sería complicado establecer.

**Diapositiva 8**

Pero antes de ponerse a diseñar (e incluso antes de analizar funcionalmente) muchas veces tener una especie de prototipo ayuda y mucho. En este caso al ser un juego básicamente de tablero, la creación de un prototipo es bastante sencillo. Y su ayuda para no perderse durante el diseño y la codificación ha sido invaluable.

Tal vez debería habérmelo traído pero dado que son 20 minutos he pensado que sería más una distracción. En cambio si que me permito la licencia de mostrar este video del como se hizo. Pensad que en parte es para añadir material desde el punto de vista de los resultados para crear hype y ventas del juego…

(Poner el video)

Y bueo hablando de ciclos y versiones (y aunque al trabajar solo pues pierde parte de…) pues es evidente la necesidad de usar software de control de versiones (he usado git).

Ya he mandado un enlace al repositorio en la introducción pero ahora me gustaría señalar esos enlaces que he puesto aquí. Representan herramientas para trabajar en equipo bastante interesantes.

La primera de proyectos donde se puede ir poniendo las tareas en paneles kanban de hechos, a medio hacer y realizados y la segunda las issues (cosas que hay que arreglar) para ir teniendo en cuenta todos los problemas y cosas a mejorar). Repito siendo un equipo de 1 no son tan útiles... Pero serán las herramientas que usaremos en el proyecto cuando seamos más

**Diapositiva 9**

¿Y el código? Bueno como he dividido el trabajo inicial en lógica, libGDX y la IA vamos a ir viendo las características más relevantes por ese orden.

La clave de la implementación del ajedrez es el siguiente diagrama (perdón porque no se vea bien en la transparencia por la falta de fondo, aunque bueno también es buscado porque lo que quiero resaltar es precisamente simplemente la estructura y relaciones entre las clases y las distintas enumeraciones creadas.

El centro de la acción es la clase Casilla que contiene aparte de sus coordenadas un Status que es los distintos estados de casilla según la peligrosidad a desaparecer y puede contener una pieza que aparte del tipo (saldrá en otro lado) será de un color. Y en esta clase Casilla ocurrirá casi todo. Es la que contiene métodos para incrementar el peligro p.e.

La clase Board contiene todas las cosas que son dependientes de la partida (las dimensiones del tablero, un arrayList de casillas, el color que le toca jugar y una lista de los colores que están jugando. Y una serie de métodos para mover las piezas en las casillas, incrementar la alerta del tablero, inicializarlo, etc.

Con respecto a las codificaciones estándar de ajedrez hay que señalar que al tener un tablero cambiante no podemos usar arrays simplemente porque no tenemos dimensiones fijas. Pero claro en un ArrayList los enlaces no son como en una matriz así que uno de los métodos principales es el que dada unas coordenadas (columna y fila) nos devuelven la casilla correspondiente. Ese es el código de la derecha. Puede parecer sencillo pero tiene su dificultad y si no llega a ser por el prototipo…

El tener que usar arraylist seguramente también nos hace menos eficientes con respecto a otros códigos para la ia pero bueno…

Por otra parte me gustaría señalar que la función que incrementa el peligro simplemente es un switch que va llamando a numero de filas según el status de la casilla de índice “cero” que básicamente es una de las esquinas del tablero. Este truco también se usa durante los movimientos de las distintas piezas para saber los límites del tablero donde se puede mover las piezas.

**Diapositiva 10**

Cuando el peligro aumenta la anterior función llama a esta con el número de filas afectadas… Y perdonadme el pseudocódigo para simplificar esta llama al método en casilla que le aumenta el stauts y devuelve true si ha eliminado la casilla… entonces hay que reducir el tamaño del tablero lo que se consigue con el if que tiene en memoria el número de filas y columnas actuales.

**Diapositiva 11**

Las piezas se ha diseñado en vista a poder ampliar a piezas de fantasía mediante una clase abstracta. Basicamente toda pieza tiene que tener un tipo de pieza y un color y dos métodos que indican que casillas se puede mover en el tablero y que casillas puede matar. En algunas piezas esto no sería necesario porque matan y mueven de la misma manera pero en el caso del peón y del rey(en este caso porque no puede mover donde le pueden matar) esto no es así asi que se requería esto.

También me gustaría resaltar el método CasillaNoDisponible que básicamente nos devuelve hasta cuando seguir con los bucles en las piezas mayores y tiene en cuenta si la casilla esta ocupada y por quien para que se pueda mover o no allí la pieza.

No voy a detenerme con los movimientos de cada pieza ni con otras peculiaridades de la codificación por cuestiones de tiempo. En la documentación espero que esté mejor explicado y siempre podeis preguntarme aquello que no esté claro.

**Diapositiva 12**

En cuanto al diseño visual. De momento he establecido 2 pantallas (una de configuración y una de juego en sí). Aquí teneis la de configuración.

La de configuración es más o menos una aplicación estándar de Scene2D.ui usando una skin del repositorio básico de skins de libgdx para hacer botones, checkbox, etc (no he usado bien los containers pero porque la skin en cuestión me lo hacía bastante feo las tablas y demás y he preferido esta estética).

En cambio para el juego en sí. Al no ser una interface normal lo que he usado es simplemente con el esquema de Scene2D un stage donde he ido añadiendo los distintos actores (el tablero modificando la clase correspondientemente y las casillas de igual manera). De esta manera cuando se dibuja el stage se transfiere el input processor y se dibujan los actores que a su vez están llevando la lógica del ajedrez por dentro.

Para esta parte he puesto mis propios assets con el TexturePackager y el atlas correspondiente que por cierto se llevan regulin con las skins. Sé que debería haber hecho una skin para esto pero… ya sabéis…

**Diapositiva 13**

Y sí, ahora tal vez debería poner la pantalla de juego pero permitidme que os ponga antes una serie de transparencias sobre una cosa que no sale en la presente versión del Chess Battle Royale pero que ha costado lo suyo programarla y que espero poder compatibilizarla con la versión actual en breve… La IA.

Hay distintas formas de introducir una IA en el ajedrez (en la documentación he puesto un breve apéndice sobre la última e interesante “moda” p.e. ). Pero la manera tradicional en juegos con información perfecta es hacer el árbol de nodos de decisiones. Hay una teoría matemática que dice que en juegos de información perfecta hay una estrategia óptima y blablablá…

En fin, como juegos como el ajedrez el árbol de decisiones es muy complicado para tener el juego “resuelto” lo que se hace es hacerlo hasta una cierta profundidad y evaluar esas posiciones. Luego se va recorriendo el árbol para arriba y según elija el jugador o el adversario se va escogiendo el valor minimo o máximo. Es lo que se conoce como algoritmo minmax.

En el árbol de la transparencia… (Explicarlo.. Entre 5 y 6 el rival haría la mejor o sea la que menos puntuación y cogería el 5… entre 5 y 4 escogerías el 5 por ser mayor etc)

Os podeis imaginar que en el ChessBattleRoyale al haber más jugadores y más tablero el árbol es mucho más complejo… Por eso necesitamos más formas de simplificar el problema uno de ellos es la poda alphabeta…

Siguiendo el diagram p.e. cuando ya se tiene que hay un valor de cuatro ya no hace falta calcular más caminos porque este siempre va a ser menor que 5… asi que esa rama/nodo no se calcula… y asi pasa con todos los nodos sombreado.

**Diapositiva 14**

Todo eso que parece tan complicado en realidad es un simple algoritmo que te encuentras en la Wikipedia. Y que aunque puede parecer algo complicado de implementar para el Chess Battle Royale no lo es tanto… Basicamente es una función recursiva que se rompe con la poda o con un nodo terminal (que no haya más movimientos o que se muera….)

Para ello tenemos que implementar algo que dada una posición nos de una serie de tableros posibles y es lo que se encarga la función TablerosPosibles.

En cualquier caso recordar que el ChessBattleRoyale es más complicado que el ajedrez (más tableros salen) y que además cada uno de ellos más complicados de tratar (usamos ArrayList en vez de simplemente arrays que podríamos usar otras formas de “recorrerlo”). Esto hace que poniendo profundidades pequeñas el tiempo de elección de la IA sea mejorable.

Esto se puede intentar hacer cambiando la forma de evaluarlo (la que elegi es lógica pero..), con mejor ordenador o intentando ordenar los movimientos y la forma de verlos de la función TablerosPosibles (no descarto que sea lo más factible).

**Diapositiva 15**

Por último decir que he dejado la posibilidad de implementar distintas IAS con este esquema. Basicamente lo que habrá que implementar es la interface Evaluation, que contiene tanto el algoritmo de la poda (por si acaso en alguna IA se quiere usar otro algoritmo) el algoritmo de evaluaciones que básicamente lo hace dependiendo del valor de la pieza, el factor de distancia a la zona de tablero que se está cayendo y cuanto de activa es una pieza.

Esta interface también se puede mejorar y ajustar de alguna automáticamente los parámetros para sacar mejores IAs.

Pero bueno todo eso es que en parte me tira mi vena antigua de buscar modelos físicos y… Porque aunque creo que el juego necesita una IA para jugar en singleplayer también sé que la parte que atraerá más al principio es el multiplayer…

**Diapositiva 16**

Bueno y por fin llegamos a las conclusiones del proyecto y cuales han sido los resultados.

En la memoria del proyecto intento ponerlas mejor pero aquí lo que he pensado es que para que malgastar saliva cuando es mucho más sencillo que lo que he hecho hable por si mismo. Y eso nos lleva a que aunque yo haya hecho muchas pruebas … yo jugando contra yo no es una partida así que esto es un momento histórico… la primera partida de ChessBattleRoyale de la historia.