Facultade de Informática da Universidade de A Coruña
Computación

PROYECTO DE FIN DE CARRERA
Ingeniería Informática

Desarrollo de un videojuego roguelike para invidentes aplicando técnicas de Procesamiento del Lenguaje Natural.

Alumno: Darío Penas Sabín
Director: Jesús Vilares Ferro
Director: Carlos Gómez Rodríguez
Fecha: CHANGE: 15 de junio de 2016

Resumen:

La industria del entretenimiento digital ha crecido inmensamente en los últimos años, llegando a alcanzar números de ventas jamás vistos anteriormente. Parte de la razón de este crecimiento viene dada por una mejora radical en el aspecto visual, necesaria para que el jugador se sienta inmerso en la aventura que se le está planteando. Estas mejoras, sin embargo, dejan de lado a muchos jugadores que, por diferentes motivos, no son capaces de apreciar el contenido visual que se les ofrece o tienen problemas para ello, haciendo imposible su disfrute.

Este proyecto consiste en la creación de un videojuego *roguelike* para invidentes que, desde un principio, parte de la idea de generar contenido específicamente diseñado para que pueda ser jugado por todo el mundo, haciendo énfasis en ofrecer al jugador una diversa cantidad de frases que describan lo que está sucediendo en su alrededor y que serán generadas automáticamente en base a las gramáticas y diccionario dados.

Para llegar a una mayor cantidad de usuarios y que su ejecución sea sencilla en el mayor número de plantaformas posible, se ha decidido usar el lenguaje de programación Java.

Lista de palabras clave:

- Tiflotecnología
- Lenguajes Naturales
- Accesibilidad
- Entretenimiento Digital
- Roguelike
- Java
- Open Source

Agradecimientos

TODO

Darío Penas Sabín
Amsterdam, PONER FECHA

Índice general

Índice de figuras vii

1. Introdución 1

1.1. Videojuegos y personas invidentes . . . 1

1.1.1. Visita a la *ONCE* . . . 2

1.2. Motivación . . . 2

1.3. Estrutura da memoria . . . 2

2. Estado del Arte 5

2.1. La industria del entretenimiento digital en la actualidad . . . 5

2.1.1. La industria, en números . . . 6

2.1.2. Conclusión . . . 6

2.2. *Roguelikes.* . . . 6

2.2.1. Qué es y orígenes . . . 6

2.2.2. En la actualidad . . . 8

2.2.3. Elementos *roguelike* en nuestro proyecto . . . 9

3. Fundamentos Tecnológicos 11

3.1. Herramientas empleadas . . . 11

3.2. Bibliotecas empregadas . . . 13

4. Metodología 15

4.1. Desarrollo en cascada . . . 16

4.2. Scrum 16
4.2.1. Prácticas recomendadas de Scrum 16
4.2.2. Valores de Scrum 16
4.3. Otras consideraciones 16
4.4. Metodoloxía seguida 16

5. Planificación y Seguimiento **17**

6. Análisis de Requisitos **19**
6.1. Consultas con la comunidad 19
6.1.1. Resumen 19
6.2. Análisis de otros elementos 19
6.3. Requisitos del aplicativo 19

7. Diseño e Implementación **21**

8. Recepción, conclusión y trabajo futuro. **23**
8.1. Recepción y *Feedback* 23
8.2. Conclusión 23
8.3. Trabajo Futuro 23

A. Escoger la licencia **25**

B. Instalación e Instrucciones **27**

Índice de figuras

2.1. Captura de pantalla de dominio público (tal y como todas las imágenes mostradas en este proyecto) del videojuego Rogue 7

2.2. Captura de pantalla del videojuego Vultures 8

Capítulo 1

Introdución

En este capítulo introductorio se explicarán los aspectos necesarios para entender lo más importante del proyecto, la motivación para realización del mismo y un breve resumen del resto de capítulos que forman parte de la memoria.

1.1. Videojuegos y personas invidentes

La mayor parte de los videojuegos comerciales no tienen en cuenta a muchas minorías de la sociedad. Haciendo una pequeña búsqueda online pueden encontrarse miles de personas quejándose de *first person shooters* que tienen un *FOV*[1] limitado, causándoles mareos al poco rato; daltónicos protestando que diferentes juegos (como por ejemplo The Witness[2]), basan buena parte de su mecánica en que el jugador sea capaz de distinguir diferentes colores; zurdos que tienen que acomodarse a ciertos controles a no existir una opción para cambiarlos; invidentes que no pueden disfrutar de prácticamente ninguno de estos títulos, etc.

En este proyecto nuestro objetivo es crear un videojuego desde cero que tenga en cuenta todo tipo de minorías, tomando especial relevancia los invidentes y centrándose en los aspectos que sean relevantes para ellos; descripciones que sean fácilmente reproducibles que cambien automáticamente y fácil expansión del título, tanto en características generales como en idiomas, gramáticas o palabras empleadas en el mismo.

[1] *Field of view*, campo de visión. extensión de mundo observable en un momento dado
[2] Juego de puzzles en primera persona: http://the-witness.net/

1.1.1. Visita a la *ONCE*

A principios de 2015 Jesús y yo fuimos a un taller de la ONCE donde una persona invidente nos habló sobre la tiflotecnología, nos mostró la forma en la que usaban ordenadores y móbiles, incluso para leer código, y los errores, muchos de ellos fácilmente solventables, que se cometían día a día en temas de accesibilidad. Esta visita nos abrió los ojos y gracias a ella fuimos capaces de detectar posibles mejoras que hacer al proyecto y los fallos que no deberíamos de cometer. También se ofrecieron a probar el proyecto una vez estuviera listo, pero este será un tema que trataremos en las próximas secciones.

1.2. Motivación

El entretenimiento digital siempre ha sido parte de mi vida y una de las razones por las que desde pequeño estuve interesado en la informática, razón por la que, finalmente, acabé estudiando esta carrera. Del mismo modo, y habiéndome relacionado con bastante gente con toda clase de necesidades especiales durante un gran periodo de tiempo, mi interés por la tiflotecnología y las limitaciones que la tecnología ofrece a millones de personas no hizo más que crecer año a año.

A pesar de los grandes avances de la industria de los videojuegos y de la gran cantidad de nuevos estudios y proyectos que se lanzan anualmente, resulta muy complicado poderse dedicar profesionalmente a cualquiera de estas dos cosas (y no digamos ambas a la vez), por lo que en mi futuro profesional no he sido capaz, al menos de momento, de cumplir mi sueño de trabajar en lo que más me apasiona. Por este motivo, cuando Jesús me comentó que él y Carlos llevaban un tiempo con este proyecto disponible, no dudé en un instante en aceptarlo y ponerme manos a la obra.

Este videojuego se ha desarrollado durante el periodo de un par de años en los que se incluyen muchos cambios en mi vida, tales como mi emigración a Holanda hace ya casi dos años y mis primeros pasos en el mundo laboral. Cada día que pasa me alegro más de tener un proyecto como éste, dado que sin la pasión e interés por el mismo estoy seguro de que jamás lo habría terminado.

1.3. Estrutura da memoria

La memoria está formada de ocho capítulos en los que se explican los pasos tomados a la hora de crear el *roguelike*.

Capítulo 1. Introducción. Se explicarán, de manera general y resumida, en qué consiste el proyecto, la motivación del mismo y la estructura que tendrá la memoria.

Capítulo 2. Estado del arte. En este apartado hablaremos de otros proyectos similares, de la situación actual de la industria sobre el problema que tratamos en este proyecto y del diferente software que es utilizado en relación con la tiflotecnología.

Capítulo 3. Fundamentos Tecnológicos. Citaremos y hablaremos sobre las herramientas y bibliotecas empleadas durante la elaboración del proyecto.

Capítulo 4. Metodología. Se detallarán las prácticas y metodologías de desarrollo empleadas para la realización del videojuego y la razón para su uso.

Capítulo 5. Planificación y Seguimento. Detallaremos la planificación y seguimiento usados en cada una de las etapas del proyecto.

Capítulo 6. Análisis de requisitos. Se comentará el análisis de requisitos para este proyecto y se explicará en detalle cada uno de los mismos.

Capítulo 7. Diseño e implementación. En este capítulo explicaremos los detalles del diseño y de la implementación de ciertas partes del programa.

Capítulo 8. Recepción, conclusión y trabajo futuro. En este apartado mostraremos los comentarios obtenidos por la comunidad durante el desarrollo del proyecto, se relatarán las conclusiones obtenidas y se detallarán los posibles cambios, mejoras y añadidos que se podrán tener en cuenta en el futuro.

Capítulo 2

Estado del Arte

En este capítulo hablaremos, principalmente, sobre dos importantes aspectos. El primero es la situación de la industria del entretenimiento digital hoy en día y qué es un *roguelike*. El segundo son los elementos que dificultan y facilitan el uso de diferentes programas y *software* a ciertos sectores de la sociedad como invidentes o daltónicos, cómo algunso programas intentan solventar estos problemas, qué opciones de accesibilidad existen en diferentes sistemas operativos y las razones por las que hemos elegido ciertas de las características de las que hemos dotado a nuestro proyecto.

2.1. La industria del entretenimiento digital en la actualidad

Desde sus primeros pasos hasta hoy en día, tal y como sucede con muchas de las novedades en el mundo del entretenimiento y la cultura, el sector del ocio digital ha sufrido cierto estigma por una gran parte de la población, siendo censurado y degragado en mayor o menor medida, no tan solo por cierta parte de la sociedad, pero también por muchos medios de comunicación y gobiernos. A pesar de que hoy en día este problema todavía está activo[1], la industria se ha expandido tanto (consolas, ordenadores, navegadores, Facebook, móvil...), que cada vez es más complicado encontrar a alguien que no haya jugado a algún videojuego en las últimas semanas y ya es algo que forma parte del día a día de mucha parte de la población.

[1] Es común que cada año en Australia se censuren algunos juegos como Paranautical Activity por razones que otras formas de entretenimiento y cultura como películas o libros no se ven tan afectados.

2.1.1. La industria, en números

En todo el mundo, pero especialmente en EEUU, la industria de los videojuegos es uno de los sectores con más crecimiento[2] llegando a generar, solamente en ventas digitales, alrededor de 61 billones de dólares en el año 2015[3].

Este gran éxito se debe, en gran parte, a la irrupción de los juegos desarrollados para móviles, cuyo beneficio ha ido aumentando enormemente durante los últimos años. [4]. Sin embargo, esto no significa que el resto de plataformas no estén triunfando. Solamente Steam, la plataforma de distribución digital para PC por excelencia desarrollada por Valve, ha generado alrededor de 3 billones y medio de dólares en el año 2015[5].

Con el mercado del PC resurgiendo, las consolas de sobremesa obteniendo grandes números de ventas, las portátiles resistiendo, el mercado de los videojuegos para móvil en esplendor y los cascos de realidad virtual llegando al mercado este año 2016; todo parece indicar que estos números no harán más que crecer en los próximos años.

2.1.2. Conclusión

Lo que comenzó hace varias décadas como un modo de entretenimiento sin ninguna pretensión, generalmente enfocado a adolescentes y que miraba a otras industrias como la cinematográfica con recelo, se ha convertido en todo lo que había deseado y más. Gracias a grandes títulos y a su expansión a toda clase de dispositivos, no se puede hablar de la industria del entretenimiento sin hablar de videojuegos y en muchos casos algunos de esos títulos han logrado ser nombrados como obras de arte en su género, pasando a la historia y siendo recordados a lo largo de los años.

Sin embargo, es bastante probable que lo mejor esté todavía por llegar.

2.2. *Roguelikes.*

2.2.1. Qué es y orígenes

En 1983, Michael Toy y Glenn Wichman crearon un videojuego llamado Rogue[6] que acabó definiendo un género.

[2] http://www.theesa.com/wp-content/uploads/2014/11/Games_Economy-11-4-14.pdf
[3] http://goo.gl/BgGhXy
[4] La venta de videojuegos en Alemania crece año tras año, pero el mayor aumento de beneficio se está centrando en el mercado de los juegos para móvil
[5] http://goo.gl/Mbjgol
[6] Desde 2014 este juego se encuentra disponible en archive.org

Las características principales que definieron a Rogue y que, por extensión, definieron al género de los *roguelikes* inicialmente, son:

Dificultad : Rogue es un videojuego difícil con *permadeath*[7] que obligará al jugador a rejugarlo una y otra vez, intentando llegar más lejos que la anterior partida gracias a ir aprendiendo los funcionamientos del mismo.

Aleatoriedad : Cada vez que el jugador comienza una partida nueva se encontrará con ciertos elementos que han cambiado con respecto a la anterior: el mapa es diferente, los elementos y enemigos se encuentran en sitios distintios, los propios objetos han cambiado... causando que cada vez que el usuario empiece, tenga un grado de dificultad pseudo-aleatorio dependiendo de la semilla con la que estos elementos han sido generados.

Progresión : Una de las frases más escuchadas en las críticas que Rogue recibió tras su lanzamiento es que el jugador sentía la necesidad de intentar llegar más lejos en cada ocasión[8]. Esto viene dado, sobre todo, por la sensación de progresión y de que en cada *run*[9] el usuario va mejorando.

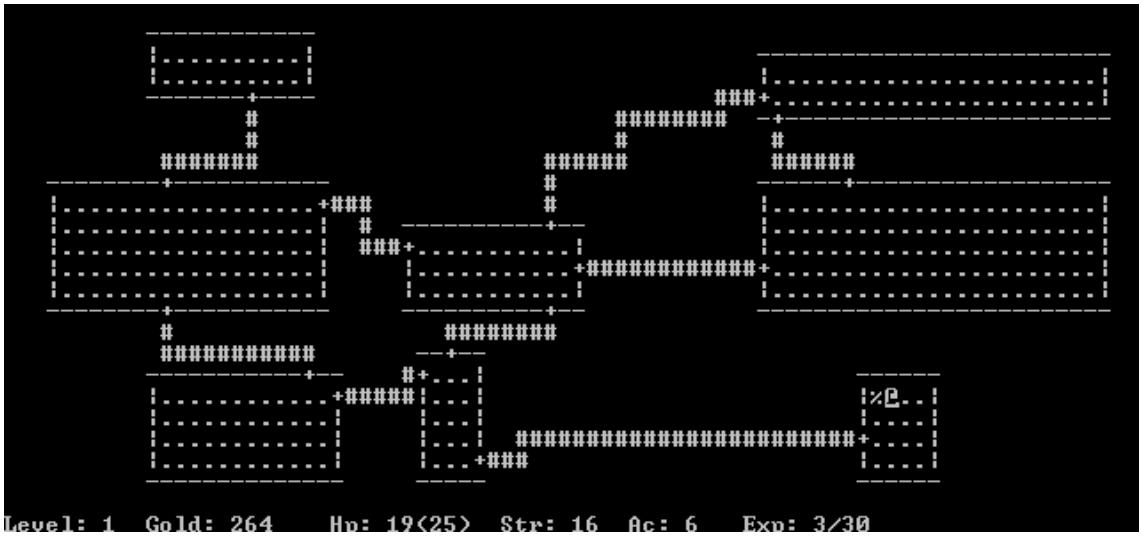

Figura 2.1: Captura de pantalla de dominio público (tal y como todas las imágenes mostradas en este proyecto) del videojuego Rogue

A partir de este momento muchos fueron los juegos que decidieron imitar estas características de Rogue, solamente cambiando diferentes elementos, por eso se han denominado *roguelikes*.

[7] Una vez que el jugador muere, tiene que empezar desde el principio; no hay partidas guardadas

[8] Jerry Pournelle habló de ello en este artículo

[9] Palabra comúnmente usada en estos géneros y que se refiere a una partida desde su inicio hasta que el jugador pierde

2.2.2. En la actualidad

Tras el éxito de Rogue, fueron muchos los títulos que simularon su fórmula de éxito e intentaron mejorarlo, sobre todo gráficamente. Muchos de ellos se centran en diferentes elementos (combate en vez de exploración, por ejemplo) y llegan a ser completamente diferentes a la hora de jugarlos (por turnos o tiempo real) pero, sin embargo, todos conservan buena parte de las características que hicieron al género famoso hasta hoy en día, al menos muchos de ellos.

Figura 2.2: Captura de pantalla del videojuego Vultures

2.2.2.1. La creación de subgéneros

Dado que perder todo el progreso y tener que empezar desde el principio sin haber conseguido nada más que la experiencia personal es algo que no atrae a mucha gente hoy en día, son numerosos los juegos que han añadido más elementos de progreso general para que el jugador no se sienta frustrado. Estos elementos pueden ser nuevos personajes con los que jugar, puntos de experiencia o dinero con lo que poder equipar y mejorar desde un principio a nuestro personaje para poder llegar más lejos que la anterior vez. También

es común ver juegos que se basan en partidas cortas, de como mucho una hora, para que la repetición sea continua y parte de la experiencia en vez de un "castigo" que tener que afrontar.

Estos cambios que se han realizado durante los últimos años y que, de cierta manera, han modificado el género que Rogue creó en un inicio, no siempre se han tomado positivamente por parte de la comunidad, quejándose de que muchos títulos que se definen a sí mismos como *roguelike* no contienen ciertos elementos, como la dificultad, que una vez definieron el género. Por este motivo se han definido subgéneros como el *roguelite*, que toman muchas de esas características, pero añaden o ignoran otras muchas para crear un título que sea un poco más sencillo y no penalice tanto al jugador.

2.2.3. Elementos *roguelike* en nuestro proyecto

En nuestro caso hemos creado un *roguelike* similar a Rogue, no solamente estéticamente, pero también en diseño y funcionalidad. El usuario se moverá por un mapa aleatoriamente generado y luchará contra diferentes enemigos que intentarán eliminarlo de diferentes formas.

El objectivo del juego es llegar lo más lejos posible dentro de la mazmorra. Cada vez que el jugador entra en un portal se añadirá un punto (el número de puntos se mostrará en la pantalla) y, cada vez que esto suceda, un nuevo mapa, con diferentes características y contenido, será generado. El juego es complicado, aleatorio y con una sensación de progreso, tal y como el género *roguelike* especifica.

INTRODUCIR IMAGEN DE NUESTRO JUEGO

Capítulo 3

Fundamentos Tecnológicos

En este capítulo hablaremos sobre los fundamentos tecnológicos que vamos a usar en este proyecto y, si cabe, la razón por la que fueron elegidas. En primer lugar citaremos las herramientas que hemos usado y, en segundo lugar, las bibliotecas que hemos decidido utilizar en el programa en sí.

3.1. Herramientas empleadas

Java Lenguaje de programación orientado a objetos cuya primera aparición fue en 1995. Es uno de los lenguajes de programación más utilizados en la industria y una de sus principales características es que es multiplataforma, es decir, puede ser ejecutado en cualquier sistema operativo que tenga la *Java Virtual Machine* instalada sin necesidad de realizar cambios en el código (WORA[1]). Esta ventaja es esencial en nuestro caso, dado que la mayoría de las personas que pueden estar interesadas en el proyecto usan una gran variedad de sistemas operativos.

Eclipse Es un IDE[2] usado para escribir código en múltiples idiomas. También incluye una serie de *plugins* que facilitan y automatizan muchas de las labores a realizar como el uso de sistema de controles, ejecución de código y tests, herramientas de debug, autocompletado de código, etc.

Git Sistema de control de versiones distribuido introducido en 2005 y desarrollado principalmente por Linus Torvalds. Es el control de versiones referencia en la mayoría de

[1] *Write once, run anywhere.* Eslogan creado por Sun Microsystems para mostrar los beneficios de la multiplataforma

[2] *Integrated Development Environment.* Entorno de desarrollo integrado

empresas y proyectos de software libre gracias a su rapidez y, al ser distribuida, permite trabajar y realizar *commits* del código sin necesidad de conexión a internet.

GitHub Plataforma de desarrollo colaborativo usada para alojar proyectos usando el sistema de control de versiones Git. La mayoría de proyectos de código abierto lo usan, dado que es gratuito, aunque también tiene la opción de almacenar el código de forma privada tras, previamente, realizar un pago.

Listas de Correo Las listas de correo son un método de comunicación muy usado por diferentes comunidades, especialmente en el desarrollo de software, que ayudan a los usuarios que participan en ellas a enviar correos a múltiples personas que lo deseen de forma anónima y, al mismo tiempo, tener un historial de las respuestas dadas por los mismos. En nuestro caso la hemos usado para comunicarnos con un grupo de usuarios y desarrolladores de videojuegos para invidentes.

Reddit Web creada en 2005 y que actualmente se encuentra en el top 50 de las más visitadas del mundo. Cuenta con una comunidad gigante que está dividida en muchísimos subgrupos dependiendo del tema a tratar. La hemos usado como una herramienta de feedback. Especialmente los *subreddits* de daltónicos`https://www.reddit.com/r/ColorBlind/` y gente ciega`https://www.reddit.com/r/blind/`.

Dia Aplicación informática que permite la creación de todo tipo de diagramas. En nuestro caso lo hemos usado para crear los diagramas UML que se encuentran en esta memoria.

JSON JavaScript Object Notation. Es un formato muy usado en APIs para intercambio de datos, similar a XML. En nuestro caso lo usamos para definir las gramáticas y diccionarios de nuestro proyecto, dado que es muy sencillo de leer y especificar. Hay numerosas bibliotecas que nos permiten analizar y trabajar con este formato en Java. La que nosotros usamos es *Gson*.

LaTeX Sistema de composición de textos altamente usado por la mayoría de textos científicos dada la facilidad de su composición, simpleza, alta calidad y herramientas que a ayudan a la creación de fórmulas, inserción de imágenes y muchos otros elementos. Muy modificable. Es el sistema que hemos usado para la creación de este documento.

NVDA Lector de pantalla de código libre para Windows. Orca es, en cierta medida, su equivalente en Linux.

3.2. Bibliotecas empregadas

Gson Biblioteca usada para transformar archivos JSON a objetos de Java y viceversa.

JCurses JCurses[3] es una biblioteca para el desarrollo de aplicaciones de terminal para JAVA. Es similar a AWT[4], pero basada en el sistema de ventanas Curses de UNIX.

Libjcsi Biblioteca de representación gráfica que trabaja sobre JCurses y simplifica la tarea de representar y refrescar elementos del terminal.

[3] *The Java Curses Library*
[4] *Abstract Window Toolkit.* Kit de herramientas de interfaz de usuario de la plataforma original de Java

Capítulo 4

Metodología

En este apartado describiremos la metodología llevada a cabo en el proyecto. Una metodología es un conjunto de procesos, métodos y prácticas llevadas a cabo para asegurar, en la mayor medida posible, calidad en el producto final y en el tiempo acordado.

En nuestro caso, al ser un proyecto realizado por una sola persona y con un tiempo diario muy limitado, hemos optado por adaptar una serie de ideas y valores principales de varias metodologías.

4.1. Desarrollo en cascada

4.2. Scrum

4.2.1. Prácticas recomendadas de Scrum

4.2.2. Valores de Scrum

4.2.2.1. Concentración

4.2.2.2. Coraje

4.2.2.3. Compromiso

4.2.2.4. Sinceridad

4.2.2.5. Respeto

4.3. Otras consideraciones

4.4. Metodoloxía seguida

Capítulo 5

Planificación y Seguimiento

Capítulo 6

Análisis de Requisitos globales

En este capítulos explicaremos el proceso de análisis de requisitos llevados a cabo para la elaboración de la aplicación y toda la información recibida por la comunidad.

6.1. Consultas con la comunidad

Blabla Blabla

6.1.1. Resumen

Blabla Blabla

6.2. Análisis de otros elementos

Otros proyectos de accesbilidad o juegos.

6.3. Requisitos del aplicativo

Con toda la información obtenida y pensada, creamos una lista con los casos de uso que nuestro proyecto debe de cumplir.

- **Whatever** Blabla

FIGURA DE CASOS DE USO UML. Hacer referencia.

Capítulo 7

Diseño e Implementación

En este capítulo mostraremos los detalles del dise;o y la implementacion de diferentes partes del proyecto.

Capítulo 8

Recepción, conclusión y trabajo futuro.

En este último capítulo detallaremos la recepción y el *feedback* recibido tras mostrar nuestro producto a la comunidad, la conclusión sacadas de la elaboración del proyecto y el posible trabajo futuro del mismo.

8.1. Recepción y *Feedback*

8.2. Conclusión

8.3. Trabajo Futuro

Blabla. Blabla

Apéndice A

Escoger la licencia

Apéndice B

Instalación e Instrucciones

El *roguelike* es software libre y el código fuente se puede encontrar en Github[1], al igual que varias versiones ejecutables. Dichas versiones ejecutables contienen un archivo .jar que solamente requiere tener una versión de JRE[2][3] superior a la 6 instalada en el sistema a usar.

Para su ejecución basta con hacer doble clic en el archivo .jar o ir al directorio donde se encuentre dicho archivo e introducir:

Fragmento de Código B.1: Comando para la ejecución del videojuego

```
java -jar game.jar
```

Para cambiar el idioma del proyecto se debe de localizar el archivo languages.properties y cambiar el idioma a ES, GL o EN para obtenerlo en español, gallego o inglés, respectivamente. También es posible cambiar las teclas del propio juego. Las que vienen por defecto se puede encontrar en la wiki del proyecto.[4]

[1] https://github.com/dpenas/roomsgame
[2] http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
[3] Java Runtime Environment
[4] https://github.com/dpenas/roomsgame/wiki