Instituto Tecnológico de Costa Rica - ITCR

Sede de Alajuela

Curso:

Lenguajes de programación IC4700

Profesora:

Samantha Ramijan Carmiol

I Semestre del 2021

Integrantes del grupo:

Jose Alexander Artavia Quesada	2015098028
Bryan Andrey Díaz Barrientos	2019264426
Josué Gerardo Gutiérrez Mora	2018300436

Diseño de la solución

El presente proyecto tiene cómo finalidad el modelado del juego *OutFoxed!* sólo que elaborado de una manera personalizada por parte de los estudiantes desarrolladores. La versión aquí presente, a diferencia del juego original, muestra cómo los personajes son todos de una época medieval, existiendo así un rey, un bufón, una princesa, un espíritu del bosque, etc. Además, cada uno de estos se diferencia de los otros por medio de las armas que posee, son estos los elementos que serán sometidos a revisión por parte de Prolog con la intención de ir descubriendo quién es el ladrón.

Hechos

El elemento distintivo de los personajes son las armas que estos poseen, estas serán las encargadas de ayudar al jugador a ver quien podría llegar a ser el ladrón. En total hay 13 armas diferentes.

```
weapon(sword).
weapon(key).
weapon(spear).
weapon(shuriken).
weapon(mullet).
weapon(axe).
weapon(flail).
weapon(gem).
weapon(crossbow).
weapon(dreamCatcher).
weapon(wand).
weapon(bow).
weapon(potion).
```

Por otra parte, otros *hechos* implementados en la elaboración de este trabajo fueron la creación de los personajes, los cuales, además de tener su propio nombre descriptivo según su posición en la sociedad medieval, se les otorga 3 armas a cada uno que servirán cómo los elementos que los diferencien de los demás para encontrar al culpable de este juego. Cada personaje tiene también un ID, que es un número entero al inicio de la especificación del hecho, esto con el fin de hacer más fácil la solución del programa para el programador.

```
character(1,aristocrate,
    weapon(potion),
    weapon(wand),
    weapon(shuriken)).
```

Inferencias lógicas

En una sola función no hay inferencias lógicas directas debido al tamaño del proyecto, por ello mismo es que la mejor manera y más ordenada de lograr llegar poco a poco a un resultado satisfactorio mediante el uso de la poderosa herramienta que es Prolog es a través del “divide y vencerás” haciendo así que, al dividir el intento de solución en diferentes funciones con trabajos diferentes pero relacionados se ha podido ir dando la creación de una inferencia lógica que dirige al programa hacia una respuesta concisa. Las partes de dicha inferencia se ven a continuación.

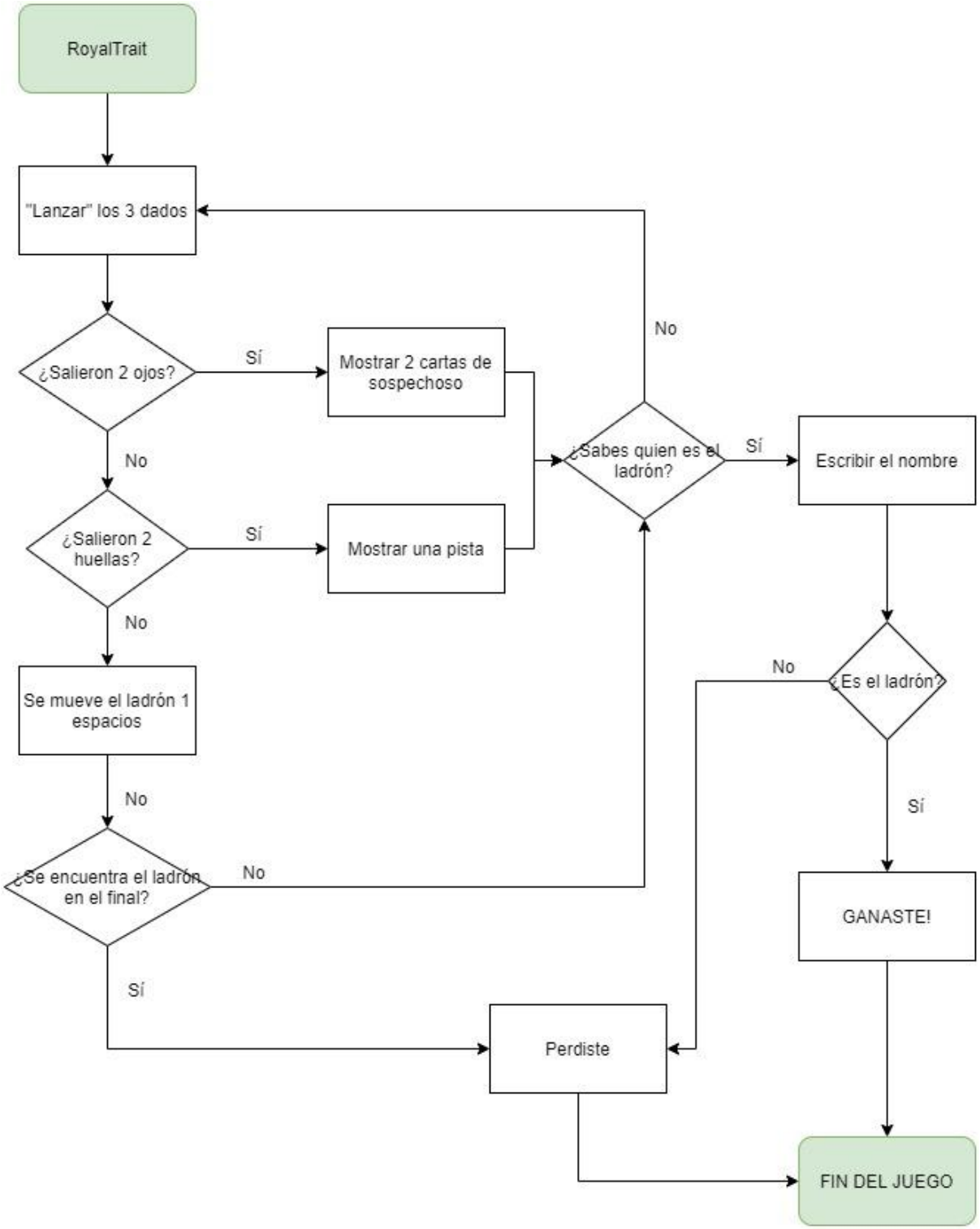
```
addToSuspects(NAME):-
    character(IDC,NAME,WC,WC2,WC3),
    assert(faceUp(IDC,NAME,WC,WC2,WC3)).

hasWeapon(Weapon):-
    guilty(_,_,weapon(Weapon),_,_);
    guilty(_,_,_,weapon(Weapon),_);
    guilty(_,_,_,_,weapon(Weapon));!.

askWeapon(Weapon):-
    hasWeapon(Weapon)->
        retract(faceUp(_,_, weapon(Weapon),_,_));
        retract(faceUp(_,_, _,weapon(Weapon),_));
        retract(faceUp(_,_, _,_,weapon(Weapon)));!.
```

askWeapon es la función principal del *decodificador* y se va encargar de llamar a la función **hasWeapon** y así ver cuales sospechosos (cartas volteadas) tienen dicha arma, encasillandolos de posibles sospechosos o en caso de no tener nada, descartando a los personajes al no marcarlos. Por otro lado, está la función **addToSuspects** que llevará el conteo de aquellos que van siendo considerados sospechosos.

Flujo del programa



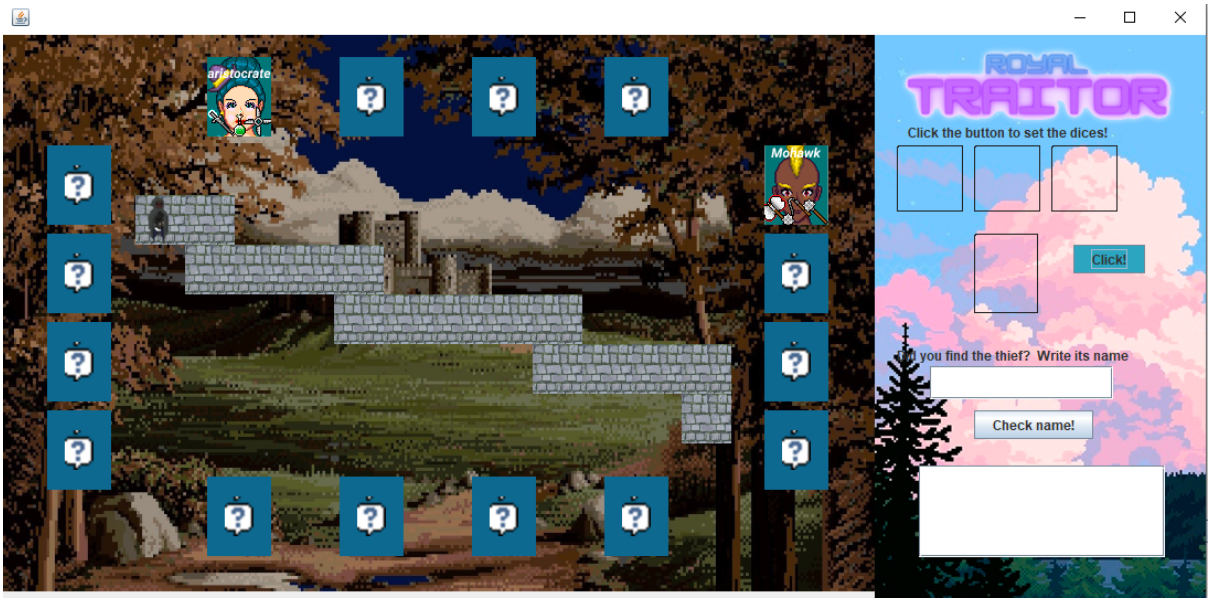
Diseño de la interfaz gráfica

Los bocetos del diseño del programa fueron generados mediante la herramienta Figma, la cual ayudaría a tener un control más amplio de cómo es que se supone que se vería la idea que se quería diseñar.

En primer lugar, se muestra el boceto de cómo se había pensado que se vería el tablero de juego en las primeras etapas.

Primer boceto de cómo se vería el tablero de juego.

Boceto final de cómo se vería el tablero de juego.



A continuación se presentan los personajes del juego y las armas, que son los elementos representativos que serán los que ayuden a saber quién fue el ladrón.

Personajes


Mohawk


princess


spirit


healer


oldman


snow


aristocrate


king


poprigunchik


child


cultist


mask


prince


clown


doll


archer

Armas


axe


bow


crossbow


dreamCatcher


flail


gem


key


mollet


potion


shuriken


spear


sword


wand

Lecciones aprendidas

1. Cómo conectar un lenguaje de programación con otro fue algo a lo que ningún miembro del equipo se había enfrentado anteriormente. Lograr esto fue toda una hazaña y se aprendió de la importancia de saber usar un entorno de desarrollo (NetBeans en este caso) no sólo para programar si no para poder realizar otras acciones, con la dicha conexión, lo cual mediante este IDE fue mucho más sencillo de lo que se pensaba.

2. La utilización de Prolog para solucionar un problema lógico, ya que, por lo general se está acostumbrado a solucionar este tipo de trabajos con programación iterativa, sin embargo, habiendo explorado un lenguaje lógico cómo lo es Prolog se pudo abordar un problema cómo el que presenta el trabajo de una manera más directa y concisa.

3. El manejo de dos lenguajes al mismo tiempo, sabiendo que lo que pasaba en un archivo de un lenguaje podía repercutir en otro de otro lenguaje, por ende había que tener un plan de trabajo claro de cómo tenía que funcionar todo en conjunto.

4. El tener cuidado con el tema de la interfaz gráfica y la parte lógica, ya que, si se iniciaba a tempranas horas a programar la interfaz esto podría ser una pérdida de tiempo ya que tal vez en la lógica algo tenía que cambiar y si eso pasaba entonces había cambios que hacer en el apartado gráfico, así que lo mejor es iniciar con lo lógico y de último la parte gráfica.

5. De la mano del punto anterior se debe de mencionar la herramienta de Figma que fue bastante útil para pensar un diseño de manera rápida y sin tener que programar nada, así cualquier cambio que se hiciera no involucró un trabajo complicado. Usar una herramienta cómo esta para modelar el aspecto gráfico fue de gran ayuda para ahorrar tiempo y a su vez saber (más o menos) a que se pretendía llegar.

Bibliografía

1. A Java interface to SWI Prolog is available using JPL. (2021). Retrieved 26 May 2021, from https://www.swi-prolog.org/FAQ/Java.html

2. how to connect a program in Prolog to a program in Java. (2021). Retrieved 26 May 2021, from https://stackoverflow.com/questions/21219997/how-to-connect-a-program-in-prolog-to-a-program-in-java

3. Bodnar, J. (2021). Java Snake game - learn how to create Snake game in Java. Retrieved 30 May 2021, from https://zetcode.com/javagames/snake/

4. Open Game Art. (2021). Most popular art of all time, Medieval Retrieved 22 May 2021, from https://opengameart.org/