***Instituto Tecnológico de Costa Rica***

***Sede Interuniversitaria de Alajuela***

***Lenguajes de Programación***

***Profesora: Samanta Ramijan Carmiol***

***I Semestre 2021***

***Integrantes:***

***-Joan Sánchez Chinchilla -> 2015123867***

***-Tomás Acuna->2018112856***

***-Marco Madrigal->2019053902***

***Tercer Proyecto Programado: Programación Lógica***

**b. Diseño de la solución:**

**b.1 Hechos**

Nuestro diseño de solución cuenta con 4 tipos de hechos diferentes, los cuales se explican a continuación.

-Personajes: Para los personajes se utilizará un identificador ,que nos brindará la posibilidad de escoger alguno de manera aleatoria por medio de una relación, el nombre correspondiente y tres accesorios y/o prendas de vestir que los caracteriza. Se vería de la siguiente manera:

personaje(Id,Nombre,Acc1,Acc2,Acc3). -> personaje(1,marco,pulsera,reloj,gorra).

-Pistas: Las pistas también utilizarán un identificador que cumple la misma función que en los hechos de personaje y el nombre correspondiente de la pista. Se vería de la siguiente manera:

pista(Id,NombrePista). -> pista(1,gorra).

-Objeto Robado: El hecho del objeto robado, al ser único en el universo del juego, simplemente contiene su nombre. Se vería de la siguiente manera:

objetoRobado(NombreObjeto). -> objetoRobado(corona).

-Dados: Los dados también contarán con un identificador, pero a diferencia de personaje y pista, este representa la cara del dado. Cada uno se asocia a lo que contiene esa cara, como por ejemplo una huella o un ojo (donde las huellas funcionan para obtener pistas y los ojos para revelar personajes). Se vería de la siguiente manera:

dado(Id,Cara). -> dado(1,huella). -> dado(2,pista).

-Puntos: Estos hechos que representarán cuantos puntos ha logrado reunir el usuario lanzando los dados. Son acumulativos y se dividen en puntos por huellas y ojos obtenidos. Cada cierto numero de puntos se le desbloqueará al usuario su respectiva recompensa como lo es obtener pistas o revelar personajes. También existen puntos para el culpable en caso de que los dados no sean del mismo tipo. Se vería de la siguiente manera:

puntosHuellas(n). -> puntosHuellas(5).

puntosOjos(n). -> puntosOjos(3).

puntosCulpable(n). ->puntosCulpable(2).

**b.2 Inferencias Lógicas**

Decidimos basar las inferencias lógicas en las pistas ya que la mayor parte del juego reside en ellas. Nuestro diseño de la solución cuenta con las siguientes:

-culpableRandom(): Esta inferencia se encargará de elegir un personaje de manera aleatoria para que sea el culpable de robar el objeto.

-quitarCulpable(): Para verificar que la inferencia anterior esté funcionado, quitarCulpable() funcionará como debug. Además, permitirá llevar un control para tener solamente un culpable por partida.

-decifradorPistas(Pista): Con esta inferencia podremos saber si una pista especifica (accesorio o prenda de vestir), incrimina al culpable.

-pistaRandom(): Esta inferencia nos brindará una pista aleatoria entre las 12 existentes.

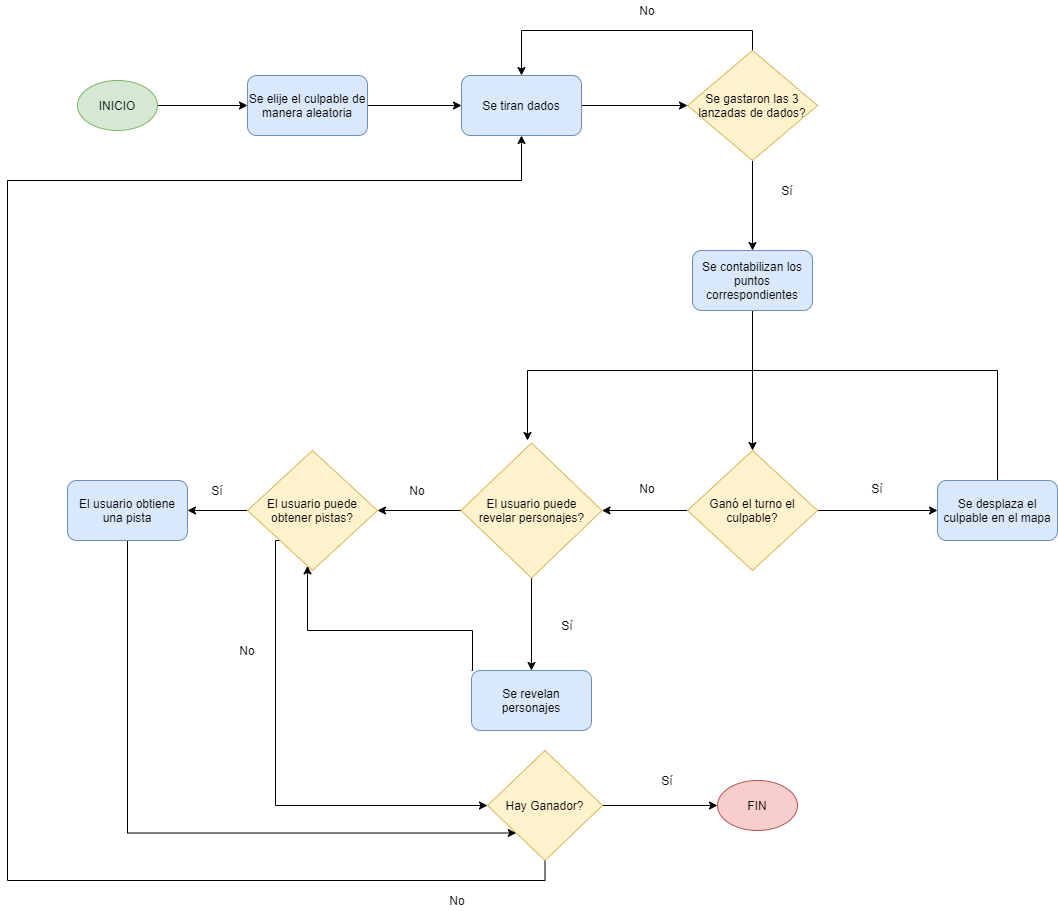
-tirarDado(): La idea de esta inferencia es poder obtener la cara de un dado luego de simular su lanzamiento, el cual puede ser producto de obtener un numero aleatorio.

-descartarPersonaje(Nombre): Con esta inferencia podremos ir descartando personajes. Además, de descartarse el culpable podremos saber si el usuario pierde la partida.

-existeGanador(): Será muy útil para saber si el culpable logró escapar o si el usuario logró atraparlo.

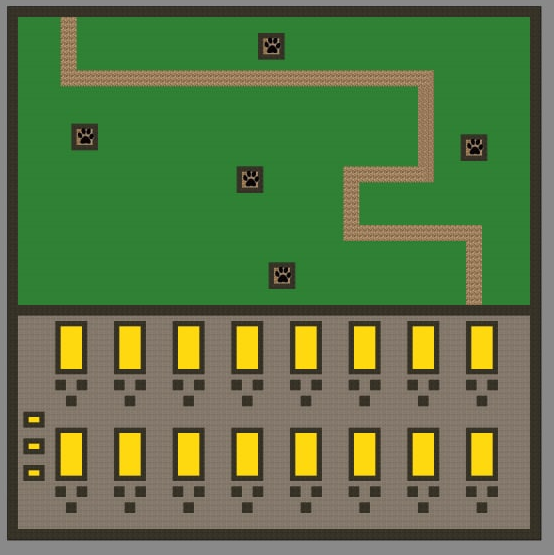
-sumarPuntosHuellas(),sumarPuntosOjos y sumarPuntosCulpable(): Con estas inferencias podremos llevar un conteo de los puntos actuales en cada ámbito.

**b.3 Flujo del programa**



**c. Diseño de la interfaz gráfica:**

Para el diseño de la interfaz gráfica decidimos realizar un boceto o representación de lo que posiblemente será el producto final del proyecto. Esta representación es la siguiente:



La parte superior corresponde al tablero del juego, donde el camino café es el camino que recorrerá el zorro. Las zonas con patas serán las zonas para las pistas, estas pueden variar en cuanto cantidad, el proyecto final podría tener más o menos de estas zonas.

En la parte inferior se encontrarán las cartas y los botones para jugar. Hay 16 cartas, por ser 16 personajes, cada carta tiene debajo 3 botones. Uno será para voltear la carta, otro para señalar que es el culpable y otro para descartar la carta (en caso de que el jugador sepa que ese personaje no es el culpable). Los 3 “botones” que se pueden observar a la izquierda, corresponderán a las acciones que se pueden realizar en el juego, como por ejemplo, revelar pistas en caso de que se pueda y tirar dados.

**d. Lecciones aprendidas:**

- Al final del proyecto no implementamos algunas inferencias que propusimos al principio como diseño de la solución. Como la de existeGanador() o las encargadas de llevar un conteo de puntos. Esto se debe a que realmente no lo vimos necesario por como se desarrolló la interfaz gráfica. Además, un sistema de puntos estaba de más y no aportaba mucho sentido a nuestra idea de juego como tal. También se corrigieron algunas o se les cambió el nombre.

-Para cumplir con los procedimientos dinámicos se aplicó un poco más el uso de retract y assertz a algunas inferencias como a la de tirarDado(), generarPistaRandom(),quitarPistaRandom(), etc.

-Hubo complicaciones con el uso de JavaFX. Ya que uno de los integrantes del grupo tenía conocimientos con Java normal manipulando JFrames, nos inclinamos por crear la interfaz gráfica de esa manera.

-A pesar de que en un principio nos parecía extraño juntar prolog con java, al final nos pareció una experiencia muy chiva. Crear un juego en base a resultados ‘true’ o ‘false’ es algo que no habíamos enfrentado antes.