Instituto Tecnológico de Costa Rica
Sede Interuniversitaria de Alajuela

SPG2019-DOC:

Documentación

Misingo

Implementación de un juego de mascotas.

Autores
Alejandro Rojas Jara
Jose Antonio Güell

15/11/2019

Tabla de Contenidos

1 INTRODUCCIÓN

1.1 Propósito

El principal objetivo de este proyecto es el de reforzar conocimientos en modelado y programación orientada objetos, mediante la implementación de un juego en Java. Se pretende que los estudiantes diseñen e implementen un juego de cuidado de mascotas, en la realización del mismo los estudiantes harán uso de los siguientes temas estudiados en clase:

1. Modelado de Clases en UML.
2. Herencia y Polimorfismo.
3. Entorno de Desarrollo (IDE) para programar en Java.
4. Programación en Java.
    a. Objetos.
    b. Clases.
    c. Atributos.
    d. Métodos.
    e. Modificadores de visibilidad.
    f. Instanciación
5. Diseño de interfaz gráfica con JavaFX.
6. Carga de datos mediante archivos JSON.

2 DESCRIPCIÓN DEL PROBLEMA

2.1 Alcance y Objetivos

Dentro del alcance del proyecto se incluye las siguientes funcionalidades y cualidades.

1. Interfaz gráfica para mejor visualización del juego.
2. El jugador podrá manejar hasta 3 mascotas diferentes simultáneamente.
3. Flujo de juego completo, es decir, se podrá jugar una partida de principio a fin.

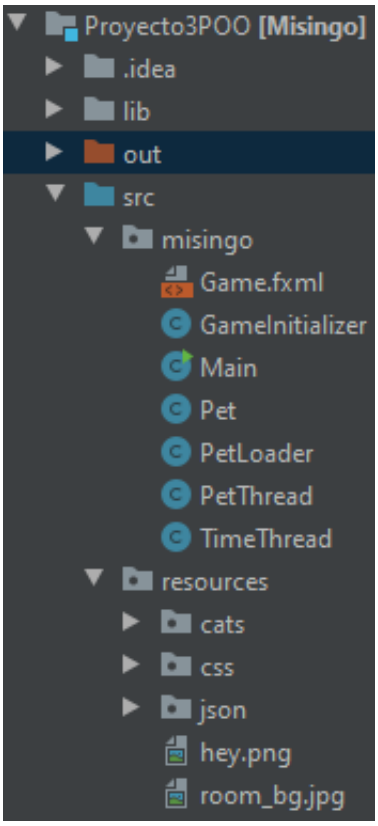
3. DISEÑO DEL PROGRAMA

3.1 Decisiones de diseño

Para este proyecto era requisito implementar interfaz gráfica, por lo que la mayoría de decisiones de diseño están basadas en base a este requerimiento.

Estructura del programa

Los archivos del proyecto se estructuraron crearon dos carpetas principales para organizar el código.



misingo: Contiene todas las clases del juego:

- **GameInitializer**: Carga e inicializa referencias a los objetos gráficos y lógicos del juego.
- **Game.fxml:** Contiene todos los atributos de los objetos gráficos, como imágenes, labels, botones, etc.
- **Pet:** Clase lógica con la información de cada mascota. Como tiempos de las necesidades y estado actual.
- **PetLoader:** Se encarga de cargar los datos de cada mascota desde un archivo JSON e instanciar cada mascota con esta información.
- **PetThread:** Hilo de ejecución para verificar las necesidades de la mascota cada hora simulada en el juego, y actualizar el estado de cada mascota.
- **TimeThread:** Hilo de ejecución para calcular y actualizar el día y hora simulada, y enviar a PetThread una señal cada vez que pasa una hora.

resources: Contiene todos las imágenes, estilos css y archivo json del juego.

3.2 Algoritmos usados

Si bien el proyecto no requirió de la implementación de algoritmos complejos, se pueden mencionar algunas funcionalidades que requirieron de la programación de cálculos lógicos para el correcto comportamiento del programa.

Día y hora simulada

Al inicio del juego, se debe iniciar un reloj virtual que muestre un tiempo “acelerado” en base al real. Específicamente, cada 5 segundos reales, se debe avanzar una hora en el juego. Consecuentemente, cada 120 segundos, se debe avanzar el día y mostrar un texto en pantalla que lo refleje. Para esto se implementó una función utilizando la clase ***Timeline,*** la cual permite realizar una ejecución de código cada segundo. Utilizando el operador modulo ***%,*** podemos determinar los múltiplos de 5, y actualizar la hora virtual cada 5 segundos. Se utilizó una lista con los días, y un índice de 0 a 7 que aumenta cada 120 segundos.

```
Timeline clock = new Timeline(new KeyFrame(Duration.ZERO, e -> {
    realSecondCounter += 1;

    //Counted 5 seconds, lets increase fake time by 1 hour
    if(realSecondCounter % 5 == 0) {
        fakeHour += 1;
        //passed 24 hours, back to 0.
        if(fakeHour == 25) fakeHour = 0;
        //An hour has passed, update Pet needs
        this.petThread.updatePetNeeds(fakeHour);
    }

    //Counted 120 seconds, a day has passed
    if(realSecondCounter == 120){
        //reset counter
        realSecondCounter = 0;
        fakeDayIndex += 1;
        //Counted to 7, back to Monday
        if(fakeDayIndex == 7) fakeDayIndex = 0;
    }
    //Set UI Labels with fake times
    this.dayLabel.setText(fakeDays[fakeDayIndex] );
    this.clockLabel.setText(fakeHour + ":00");
}),
        new KeyFrame(Duration.seconds(1))
);
clock.setCycleCount(Animation.INDEFINITE);
clock.play();
```

Imagen 1. Función del reloj virtual.

Imagen 2. Interfaz del juego completa.

4.3 Bibliotecas usadas

- La principal biblioteca utilizada es **JavaFX**, la cual permite la creación de diferentes controles como botones, imágenes, labels, usando un archivo **fxml (JavaFX XML).**
- Se utiliza **javafx.animation** por sus clases *Animation* y *Timeline*. También se hace uso de **java.util** para el manejo de **ArrayLists**.

5. ANÁLISIS DE RESULTADOS

5.1 Objetivos alcanzados

Se logró completar la implementación de los aspectos gráficos en su totalidad. Los estados de las mascotas son actualizados correctamente con el paso de las horas virtuales.

5.1 Objetivos no alcanzados

Aspectos de jugabilidad no se lograron implementar debido a falta de tiempo. No se puede alcanzar el final de la partida.

6 AUTOEVALUACIÓN

Nombre	Evaluación
Jose Antonio	100
Alejandro	100