

TP Progra 1: informe

Juego “plantas invasoras” – grupo 9

Integrantes:

-Manuel Girardi.

-Víctor Klappembach.

-Guillermo Sheil.

Profesores:

-Leonor Gutiérrez.

-Leonardo Waingarten.

Comisión: 01 (turno tarde)

Introducción: en este trabajo, nuestra tarea es hacer un juego en lenguaje java, que trata de la perra rescatista Layka debe salvar una ciudad de un grupo de plantas mutantes escupe fuego.

Nuestro principal objetivo crear un código funcional, que sea claro y fácil de comprender para quien lo vea.

Desarrollo: el proyecto se compone de las siguientes clases: Juego, Layka, Planta, Auto, RayoDestructor, BolaDeFuego, Calles, Manzana, Puntaje, Lapida, Item, Informacion Y cartel.

La clase Juego es la más importante para el proyecto. En esta se inicializan los objetos layka, los arreglos de plantas y autos, los booleanos de control, el temporizador y otras variables que les hablare después.

En el constructor del mismo se inicializan todas las variables y se les asignan valores. En particular la gran mayoría de los booleanos se inicializan en false salvo el de sigueViva que es true al inicio del juego y las variables enteras se inicializan en 0. Para asignarle las posiciones tanto a los autos como a las plantas creamos dos arreglos de enteros: direccionDeLasPlantas y direccionDeLosAutos, los cuales toman valores de dos random rV y rH, que representan las direcciones en donde estos objetos se mueven en forma vertical y horizontal respectivamente.

También están los arreglos plantasMuertas y autosRotos, los cuales van a servir más adelante, al igual que el resto de los booleanos.

Esta clase tiene también el método tick(), en esta se les las propiedad a los objetos definidos con anterioridad.

Los objetos layka, plantas y autos, si estos no son nulos, se los dibuja y se les da movimiento con sus métodos propios definidos en sus clases.

En el caso del rayoDestructor y las bolasDeFuego estos representan los ataques de layka y las plantas respectivamente.

El rayo se dibuja y cuando este no sea null se presione la tecla espacio y este se mueve en la dirección que está este apuntando. No se puede dibujarse otro rayo a menos que el que dibujo salga de la pantalla o le dé a alguna planta o bola de fuego. Que este sea null se controla con el booleano seDisparoRayo, el cual pasa a ser true cuando se pueda dibujar otro.

En el caso de las bolas de fuego estas tienen la particularidad que se dibujan con un temporizador y no siempre en la misma plantas y al igual que los rayos están se vuelven null cuando le pegan a layka o a algún auto.

Cuando una planta es golpeada por el rayo y un auto es golpeado por una bola de fuego estos pasan a ser plantasMuertas y autosRotos respectivamente. Estos tienen un timer definido como tiempoParaRevivir y tiempoParaRepararse, el cual vuelven al arreglo de plantas y autos y se vuelven a dibujar en pantalla y a moverse.

En el caso de layka esta tiene las siguientes peculiaridades. Esta tiene 3 corazones, cuando una bola de fuego la toca pierde una y si pierde las 3 muere y pasa a ser null. Mientras que si una planta o auto lo hace, esta muere automáticamente. Este puede recuperar sus corazones con los ítem generados en el centro de la pantalla, de uno a la vez.

Layka puede usar el rayo para defenderse de las plantas y bolas de fuego, mientras que no puede atacar a los autos con el pero los puede esquivar cambiando de carril.

Usando los métodos restriccionPared y restriccionManzana para que layka no salga de la pantalla y que esta solo se pueda mover por las calles respectivamente, también generando las colisiones con esos objetos en el proceso.

Cuando layka pasa a ser null, se muestra la clase Lapida, que dibuja una sprite de ella muerta en las coordenadas donde murió.

Para ganar el juego se consideran las siguientes clases. La primera representa el puntaje acumulativo generado al matar plantas el cual es de 5 por cada planta eliminada.

La clase cartel toma de base la variable el resultadoPartida, la cual mostrara victoria si esta es igual a 0 y derrota si es distinta a ese valor. Mostrando un gif animado en pantalla con esos mensajes respectivamente.

Para que resultadoPartida sea 0, el puntaje acumulado durante el juego tiene que ser como mínimo 100 y que layka siga viva.

Las clases Manzana y Calles representan al mapa del juego el cual se dibuja en el entrono con el método dibujarMapa que toma los arreglos de manzanas, callesVerticales y callesHorizontales, y los dibuja.

Calles determina qué tipo de calle se dibuja dependiendo de la dirección. Si esta es igual a 0 esta es vertical y si es igual a 1 es horizontal.

Nuestros principales problemas a la hora de desarrollar el juego fueron los siguientes.

-el generar las plantas de forma tal que puedan generarse en todas direcciones y moverse de forma adecuada por el entorno. Este se resolvió crean el arreglo de direccionesDePlantas y poniendo los random mencionados con anterioridad.

-al transformar un objeto ya sea una planta, una bola de fuego o la misma layka a null hacia que el juego generara muchos errores.

-evitar que una planta sea eliminada cuando pase en la posición original de layka, incluso cuando el rayo no estaba en pantalla. Esto se soluciono al poner en la clase layka un método disparar que actualiza el objeto rayo en las posiciones y dirección actuales de layka.

Hablando más en profundidad de las clases, las clases Planta y Auto se mueven basándose en las direcciones. Si esta es 0 se van a mover de arriba abajo, si es 1 de derecha a izquierda, si es 2 de abajo a arriba y si es 3 de izquierda a derecha.

Las clases rayoDestructor y Layka en cambio se mueven basando en el siguiente código **this**.direccion=d;

**if** (direccion ==0)

{

y-=3; 10 en rayo

}

**if** (direccion ==1)

{

x+=3; 10 en rayo

}

**if** (direccion ==2)

{

y+=3; 10 en rayo

}

**if** (direccion ==3)

{

x-=3; 10 en rayo

}

Y bolaDeFuego de la siguiente forma **if** (direccion ==0)

{

y-=velocidad\*2;

}

**if** (direccion ==1)

{

y+=velocidad\*2;

}

**if** (direccion ==2)

{

x+=velocidad\*2;

}

**if** (direccion ==3)

{

x-=velocidad\*2;

}

Conclusión: logramos crear un juego estable, entretenido y con un buen reto, que tiene un código fácil de comprender y ordenado.

Logramos cumplir con nuestro objetivo y lo que se nos cumplía en la consigna

También aprendimos con este trabajo a cómo hacer colisiones de objetos y restricciones en java, a cómo usar un entrono para desarrollar un juego y como hacer un timer.