Universidad Zaragoza
1542

Shamuel Manuel Manrrique Aquino
NIP: 802400

Tetris Agentes

Resumen

Esta práctica consiste implementar algunas funcionalidades para lograr el correcto funcionamiento del juego tetris usando la librería JADE de Java en el ambiente de desarrollo integrado Netbeans. El juego plantea la creación de dos agentes uno de tipo Juego que ofrece un servicio que atiende los eventos que contiene las peticiones de movimiento a ejecutar y validar si las acciones a ejecutar son válidas. Por otro lado el tipo controlador que es el encargado de encontrar los agentes que ofrezcan el servicio que requiere para que ejecute las acciones en el juego.

Objetivo

1. El objetivo de la práctica es lograr el correcto funcionamiento del juego. Para lograr este objetivo se debe cumplir con:
    a. Implementar funciones faltantes en frm_Controles y frm_Controles.
    b. Implementar lógica de agente controles y agente juego.

Implementación

Para el desarrollo de este proyecto se seccionó en las siguientes etapas descritas a continuación:

1. **frm_Controles:** En esta sección se crearon un GuiEvent para cada una de las acciones en este caso se consideran tres(“left, right,turn”) al usuario seleccionar alguna acción es comunicada al controlador usando postGuiEvent.

```
private void btn_IzquierdaActionPerformed(java.awt.event.ActionEvent evt) {
        //  System.out.println(" LEFT ");
        GuiEvent  evento = new GuiEvent(this, 0);
        evento.addParameter("LEFT");
        this.gui.postGuiEvent(evento);
}
private void btn_DerechaActionPerformed(java.awt.event.ActionEvent evt) {
        //  System.out.println(" RIGHT ");
        GuiEvent  evento = new GuiEvent(this, 1);
        evento.addParameter("RIGHT");
        this.gui.postGuiEvent(evento);
}
```

Shamuel Manuel Manrrique Aquino
NIP: 802400

```
private void btn_GirarActionPerformed(java.awt.event.ActionEvent evt) {
        // System.out.println(" TURN 360 ");
        GuiEvent  evento = new GuiEvent(this, 2);
        evento.addParameter("TURN");
        this.gui.postGuiEvent(evento);
}
```

2. **frm_Juego:** En esta parte se hizo un trabajo similar a la parte 1 a diferencia que aquí se maneja una sola acción que es iniciar juego. También es muy importante la creación del objeto BajarPieza que que es enviado como parámetro en el evento. Se comunica la acción postGuiEvent para notificar dicha acción a el Agente juego.

```
private void btn_Iniciar_JuegoActionPerformed(java.awt.event.ActionEvent evt) {
        System.out.println(" START GAME AGENT ");
        GuiEvent  evento = new GuiEvent(this, 0);

//        Create pieza and send to  start Juego
        BajarPieza pieza = new BajarPieza(this);
        evento.addParameter(pieza);
        this.gui.postGuiEvent(evento);
}
```

3. **Controles:** Su trabajo es informar mediante el paso de mensaje al "agente Juego" sobre las acciones que ejecuta el usuario mediante la interfaz gráfica que consta de tres botones identificados con cada una de las acciones válidas en el juego(left, right, turn). Esto se logra mediante tres funcionalidades básicas:

    a. **Crear Servicio:** se crear una solicitud con las especificaciones del servicio que se desea solicitar.

```
//  Creating Agent description
    DFAgentDescription descripcion = new DFAgentDescription();
    System.out.println("Creating service request Juego");

//  Creating service that the agent needs to request
    ServiceDescription servicio = new ServiceDescription();
    servicio.setType("Tetris");
    servicio.setName("Juego");

//  Add more information to agent description
    descripcion.addLanguages("castellano");
    descripcion.addServices(servicio);

    System.out.println("Searching service name Juego");
    this.agente.doWait(100);
```

Shamuel Manuel Manrrique Aquino
NIP: 802400

b. **Buscar agentes:** se busca en paginas amarillas un agente que brinden el servicio requerido , en este caso de tipo Juego. Se inicia el juego en caso de encontrarlo, en caso contrario se sigue buscando un agente que ofrezca el servicio.

```
//  Searching all service registered in yellow pages
DFAgentDescription[] resultados = DFService.search(this.agente, descripcion);

if (resultados.length > 0) // Found services
{
    System.out.println("Tenemos en total de " + resultados.length +" Agentes
jugando...");
//  Set True to encontrado
    this.encontrado = true;

//  I need call onGuiEvent, For is not necesary
    for (DFAgentDescription agente:resultados) {
        this.agente.agenteJuego = agente.getName();
    }

    ventana.setJugando();

} else // Didn't find services
{
    System.out.println("No existen Agentes dispuestos a jugar...");
//  Set False to encontrado
    this.encontrado = false;
    ventana.setSinJuego();
}
```

c. **Enviar mensaje:** se realiza envíos de mensajes para la comunicación de tipo ACLMessage.INFORM con el valor de la acción para notificar al agente juego la acción a ejecutar.

```
protected void onGuiEvent(GuiEvent ge) {

    if (ge.getType() == 0) // Botton Left
    {
        //  System.out.println(" Parameter 0 " + (String)ge.getParameter(0) );
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.addReceiver(this.agenteJuego);
        msg.setContent((String)ge.getParameter(0));
        send(msg);
    }else if(ge.getType() == 1) // Botton Right
    {
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.addReceiver(this.agenteJuego);
        msg.setContent((String)ge.getParameter(0));
        send(msg);
```

Shamuel Manuel Manrrique Aquino
NIP: 802400

```
 }else if(ge.getType() == 2)// Turn around
    {
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.addReceiver(this.agenteJuego);
        msg.setContent((String)ge.getParameter(0));
        send(msg);
    }
}
```

4. **Juega:** Su rol es ofrecer un servicio de tipo juego y atender las peticiones de movimientos a ejecutar en la interfaz gráfica validando si los movimientos son posibles o no. Para el correcto funcionamiento de esta última sección se desarrolló las siguientes funcionalidades básicas.

    a. **Crear servicio e iniciar juego:** Esta es la parte inicial se activa al recibir un evento del frm_juego donde e indican el inicio del juego. Luego de esto se crea un servicio indicando toda la información requerida y se publica el servicio en la plataforma. Este se conoce como Páginas amarillas según las normativas FIPA. Este mecanismo permite que otros agentes puedan solicitar dicho servicio. Una vez que el servicio esté listo se extrae el objeto de tipo pieza que viene en el evento y se inicia el hilo con pieza.start() para dar comienzo al juego. También se encarga de notificar al controlador que el juego ha terminado cuando obtenga un GuiEvent con getType() igual a auno.

```
    protected void onGuiEvent(GuiEvent ge) {
        System.out.println("Receive event  JUEGOOOO"+ge.toString());
        if (ge.getType() == 0) // Start game
        {
//          Creating Agent description
            System.out.println("Creating service Juego");
            DFAgentDescription descripcion = new DFAgentDescription();
            descripcion.setName(getAID());

//          Creating service offered by the agent
            ServiceDescription servicio = new ServiceDescription();
            servicio.setType("Tetris");
            servicio.setName("Juego");

//          Add more information to description
            descripcion.addLanguages("castellano");
            descripcion.addServices(servicio);
```

Shamuel Manuel Manrrique Aquino
NIP: 802400

```
//          Register the service in the platform
            try {
                DFService.register(this, descripcion);
            } catch (FIPAException e) {
                e.printStackTrace();
            }

            BajarPieza pieza = (BajarPieza) ge.getParameter(0);
            pieza.start();
        }else if (ge.getType() == 1) // End game
        {
            System.out.println("Termino juego, bloquear controles");
            System.out.println(this.end_game);

//            Send message to inform that the game is over
            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
            msg.addReceiver(sender);
            send(msg);
            System.out.println(msg);

        }
    }
```

b. **Validar mensaje:** Una vez recibido un mensaje se valida que no esté vacío y que sea de tipo ACLMessage.INFORM. Si cumple con esas condiciones se extrae la acción y se procede a ejecutarla.

```
ACLMessage resp = this.myAgent.receive();
if (resp != null) // Check if message if no null
{
    System.out.println(" READY TO RECEIVE MESSAGE"  );
    if (resp.getPerformative() == ACLMessage.INFORM) // Check if ACLMessage type
    {
        try {
            //  Geting action in the message
            String action = resp.getContent();
            System.out.println("Action to execute: "+ action );

            //  Execute action
            this.realizarAccion(action);
        } catch (Exception ce) {
            System.out.println("error " + ce.toString());
        }
    }
}
```

c. **Ejecutar acción:** En esta parte seleccionamos que tipo de acción se pide ejecutar, se chequea su validez y por último se ejecuta para que se vea reflejada en la interfaz gráfica.

```
private void realizarAccion(String accion) {
    if (accion.equals("LEFT")) // Check left action
    {
        if (juego.estaLibreIzquierda()) {
            Pieza.getInstancia().moverAIzquierda();
            juego.mostrarPieza();
            juego.borrarRastroHaciaIzquierda();
        }
    } else if(accion.equals("RIGHT")) // Check rignt action
    {
        if (juego.estaLibreDerecha()) {
            Pieza.getInstancia().moverADerecha();
            juego.mostrarPieza();
            juego.borrarRastroHaciaDerecha();
        }
    } else if(accion.equals("TURN")) // Check turn action
            {
                int oldPosition = Pieza.getInstancia().getEstado();
                if ( oldPosition == 1 && juego.estaLibreDerecha() ) {
                     System.out.println(" Pieza 1"  );
                    juego.borrarPieza();
                    Pieza.getInstancia().setEstado(4);
                    juego.mostrarPieza();
                }else if ( oldPosition == 2) {
                    juego.borrarPieza();
                    Pieza.getInstancia().setEstado(1);
                    juego.mostrarPieza();
                }else if ( oldPosition == 3 && juego.estaLibreIzquierda()) {
                    juego.borrarPieza();
                    Pieza.getInstancia().setEstado(2);
                    juego.mostrarPieza();
                }else if ( oldPosition == 4 && juego.estaLibreBajando()) {
                    juego.borrarPieza();
                    Pieza.getInstancia().setEstado(3);
                    juego.mostrarPieza();
                }
            }

}
```

d. Pieza aleatoria: Para hacer el juego un poco más interesante se creó la función aleatoria que da como resultado un valor de uno al cuatro, esto permite que la pieza al iniciar el juego tenga distintos estados.

Universidad Zaragoza
1542

Shamuel Manuel Manrrique Aquino
NIP: 802400

```
//  Get differents values state between 0 and 4 to start
int state = posiPieza();
setEstado(state);

public int posiPieza(){
    int min = 1;
    int max = 4;

    Random r = new Random();
    int randomNum =  r.nextInt((max - min) + 1) + min;

    return randomNum;

}
```