ABAcore Animátor

Animátor je len v testovacej fáze.

Metodická príručka

Vytvorenie a vymazanie animátora

Vytvorenie animátora

Trieda Simulation obsahuje metódy createAnimator() a setAnimator(IAnimator) na vytvorenie, nastavenie animátora. Zavolaním metódy createAnimator() sa vytvori defaultný animátor ABASimu. Používatelia si môžu vytvoriť vlastný animátor implementujúci rozhranie IAnimator a zaregistrovať ho do simulácie pomocou metódy setAnimator(IAnimator).

```
Simulation simulation = new MySimulation();
simulation.createAnimator();
simulation.setAnimator(new VlastnyAnimator());
```

Animátor ABASimu môže byť vytvorený kedykoľvek pred spustením simulácie a počas behu simulácie. Animátor ABASimu tiež ovplyvňuje rýchlosť simulácie svojim nastavením rýchlosti. Animátor ABASimu po svojom vytvorení nemá nastavenú maximálnu rýchlosť. Majte tento fakt na pamäti, aby ste po vytvorení animátora ABASimu nepovažovali spomalenú simuláciu animátorom ABASimu za chybu v programe. Vytvorenie alebo nastavenie animátora je najvhodnejšie vykonať niektorým manažérom agenta simulácie za behu simulácie.

```
public class ManagerAgenta extends Manager{
        …
        public void processMessage(MessageForm message) {
                …
                switch(message.code())
                {
                …
                case MessageCodes.vytvorAnimator:
                        …
                        mySim().createAnimator();
                        //alebo
                        mySim().setAnimator(new VlastnyAnimator());
                        …
                break;
                …
                }
                …
}
```

…
}

Vymazanie animátora

Vymazanie animátora je jednoduché stačí zavolať jednu z metód removeAnimator() alebo setAnimator(null) triedy Simulation. Vymazanie animátora je najvhodnejšie vykonávať za behu simulácie.

```
Simulation simulation = new MySimulation();
simulation.removeAnimator();
simulation.setAnimator(null);
```

Notifikácie o zmene Animátora

Sledovanie vytvorenie, odstránenie animátora je možné dvoma spôsobmi. Prvou možnosťou je priame vloženie kódu, ktoré sa má vykonať pri zmene animátora prostredníctvom lambda funkcií.

```
Simulation simulation = new MySimulation();
simulation.onAnimatorWasCreated((old, new) -> { ... });
simulation.onAnimatorWasRemoved((old) -> { ... });
```

Metóda onAnimatorWasCreated(BiConsumer<IAnimator,IAnimator>) je volaná v prípade vytvorenia, nastavenia nového animátora. Parameter old predstavuje predchádzajúci animátor. Parameter new predstavuje nový animátor.

Metóda onAnimatorWasRemoved(Consumer<IAnimator>) je volaná v prípade odstraňovania animátora volaním metódy removeAnimator() alebo setAnimator(null) triedy Simulation. Parameter old predstavuje predchádzajúci animátor.

Druhou možnosťou je registrácia animačného delegáta simulácií - návrhový vzor observer. Simulácia môže mať ľubovolný počet animačných delegátov. Animačný delegát je notifikovaný o zmenách stavu animátora a implementuje rozhranie IAnimDelegate. Registrovanie animačného delegáta sa vykoná metódou registerAnimDelegate(IAnimDelegate) objektu simulácie.

```
public class MyAnimDelegate implements IAnimDelegate {
        public void animatorCreated(IAnimator old, IAnimator new) {
                //Kód odobratia starého kanvasu z GUI
                …
                JComponent forRemove = old.canvas();
                …
                //Kód pridanie nového kanvasu do GUI
                …
                JComponent forAdd = new.canvas();
                …
```

```
        }
        public void animatorRemoved(IAnimator old) {
            //Kód odobratia starého kanvasu z GUI
            …
            JComponent forRemove = old.canvas();
            …
        }
    }
```

Metódy animatorCreated(IAnimator, IAnimator), animatorRemoved(IAnimator) sú zhodné s metódami onAnimatorWasCreated(BiConsumer<IAnimator,IAnimator>), onAnimatorWasRemoved(Consumer<IAnimator>).

Odporúča sa použiť len jeden z uvedených spôsobov. Notifikácia o zmene animátora slúži na pridanie, odobratie kanvasu animátora do, z GUI. Kanvas animátora sa získa zavolaním metódy animator().canvas() na objekte triedy Simulation. Kanvas je potomkom triedy JComponent a je grafickou reprezentáciou animátora.

Vzniknuté notifikácie o zmene animátora vykonáva simulácia počas svojho behu. To znamená, že notifikácie spôsobené vytvorením alebo odobratím animátora pred spustením simulácie budú spracované až po spustení simulácie.

Použitie animátora

Animačné objekty

Úlohou animátora je zobrazovať stav simulácie. To sa deje za použitia animačných objektov. Počas behu simulácie vytvárame, rušíme animačné objekty, s ktorými manipulujeme a tým animujeme simuláciu.

Animátor ABASimu používa 3 základné animačné objekty predstavujúce: obrázok, text a tvar. Animačný objekt obrázka je reprezentovaný triedou ImageAnimObject, animačný objekt textu je reprezentovaný triedou TextAnimObject a animačný objekt tvaru je reprezentovaný triedou ShapeAnimObject. Všetky tieto triedy majú spoločného predka triedu AnimObject. S animačnými objektmi sa pracuje v manažéroch, asistentoch a kontinuálnych asistentoch.

Registrácia animačného objektu

Aby bol animačný objekt zobrazovaný, musí sa zaregistrovať do animátora. Zaregistrovanie animačného objektu prebieha zavolaním metódy animator().registerAnimObject(IAnimObject) na objekte triedy Simulation. Animačné objekty animátora ABASimu môžu byť zaregistrované len v jednom animátore ABASimu.

Po zaregistrovaní animačného objektu do animátora, animátor ABASimu nastaví animačnému objektu id a animator do ktorého je animačný objekt zaregistrovaný. Pri pokuse o zaregistrovanie už registrovaného animačného objektu do animátora ABASimu nastane výnimka.

```
public class ManagerAgenta extends Manager{
        …
        public void processMessage(MessageForm message) {
                …
                AnimObject animObject = new ImageAnimObject(...);
                …
                mySim().animator().registerAnimObject(animObject);                    …
        }
        …
}
```

Odobratie animačného objektu z animátora

Ako náhle niektorá entita v simulácií predstavovaná animačným objektom ukončila svoju činnosť(zákazník odišiel z reštaurácie), treba odobrať animačný objekt zobrazujúci túto entitu z animátora.

Odobratie animačného objektu sa vykoná zavolaním metódy animator().removeAnimObject(long) na objekte triedy Simulation. Parameter metódy je id animačného objektu. Získanie id animačného objektu ABASimu sa vykoná zavolaním metódy id() na objekte animačného objektu. Animačné objekty ABASimu získavajú id po zaregistrovaní do animátora.

Animačné objekty animátora ABASimu poskytujú ešte ďalší spôsob odobratia animačného objektu z animátora zavolaním metódy removeFromAnimator() na objekte animačného objektu. Po odobratí animačného objektu ABASimu z animátora ABAsimu sa animačnému objektu odoberie id a animátor vlastniaci animačný objekt. Animačný objekt je možné znova registrovať.

```
public class ManagerAgenta extends Manager{
        …
        public void processMessage(MessageForm message) {
                …
                Sprava s = (Sprava) message;
                mySim().animator().removeAnimObject(s.animObject().id());
                //alebo
                s.animObject().removeFromAnimator();
                …
        }
        …
}
```

Animačné aktivity

Aby sme mohli manipulovať s registrovanými animačnými objektami, musíme vytvárať animačné aktivity a registrovať ich do animátora. Keď predá simulácia riadenie animátoru, animátor vykoná tieto aktivity.

Registrovanie animačnej aktivity je vykonané príkazom animator().registerAnimActivity(IAnimActivity) na objekte triedy Simulation.

Animačné objekty animátora ABASimu ponúkajú metódy, ktoré vytvárajú animačné aktivity a registrujú ich do animátora. Animátor ABASimu má nasledujúce animačné aktivity.

ImmediateAActivity

Okamžitá aktivita slúži na zmenu stavu animačného objektu, ako napríklad zmenenie textu textového animačného objektu, alebo zmenenie polohy animačného objektu atď. Touto triedou je možné obsiahnuť každú zmenu vykonanú na animačnom objekte bez vytvárania jej potomkov a prekrývania metódy animate(). Kód na zmenu stavu animačného objektu je možné vložiť formou lambda výrazu do konštruktora ImmediateAActivity(long, Consumer<PNode>) triedy tejto aktivity. Parameter typu long je id animačného objektu. Parameter Consumer<PNode> je lambda funkciou na zmenu stavu animačného objektu. Animačné objekty animátora ABASimu zaobaľujú objekty, ktoré sa zobrazujú na kanvase. Tieto zobrazované objekty sú potomkami triedy PNode.

Animačný objekt ImageAnimObject zaobaľuje triedu PImage, animačný objekt TextAnimObject zaobaľuje triedu PText a animačný objekt ShapeAnimObject zaobaľuje triedu PPath.

```
public class ManagerAgenta extends Manager{
        …
        public void processMessage(MessageForm message) {
                …
                Sprava s = (Sprava) message;
                AbstractAActivity act = new ImmediateAActivity(
                        s.animObject.id(),
                        (body) -> {
                                PText txtBody = (PText) body;
                                txtBody.setText(“TEXT“);
                });
                mySim().animator().registerAnimActivity(act);
                //equivalent – metóda vytvárajúca a registrujúca animačnú
//aktivitu
                ((TextAnimObject)s.animObject()).setText(“TEXT“, false);
                …
        }
        …
```

```
}
```

MoveAActivity

Animačná aktivita na premiestnenie animačného objektu z bodu A, v ktorom sa nachádza do bodu B so zadaným trvaním. Preto že táto aktivita trvá zadaný čas, je ju najlepšie použiť v kontinuálnych asistentoch.

```
public class ProcesPresunu extends ContinualAsistant{
        …
        public void processMessage(MessageForm message) {
                …
                double trvanie = _rnd.next();
                Sprava s = (Sprava) message;
                AbstractAActivity act = new MoveAActivity(
                        new Point2D.Double(100,100),
                        trvanie, s.id());
                mySim().animator().registerAnimActivity(act);
                //equivalent – metóda vytvárajúca a registrujúca animačnú
//aktivitu
                s.animObject().animateToPosition(
                        new Point2D.Double(100,100), trvanie);
                …
        }
        …
}
```

MoveScaleRotationAActivity

Animačná aktivita na premiestnenie animačného objektu z bodu A, v ktorom sa nachádza do bodu B so zadaným trvaním cieľovou rotáciou a zmenou veľkosti. Preto že táto aktivita trvá zadaný čas, je ju najlepšie použiť v kontinuálnych asistentoch.

```
public class ProcesPresunu extends ContinualAsistant{
        …
        public void processMessage(MessageForm message) {
                …
                double trvanie = _rnd.next();
                Sprava s = (Sprava) message;
                AbstractAActivity act = new MoveScaleRotationAActivity(
                        new Point2D.Double(100,100),
                        1.5d, 0d, trvanie, s.id());
                mySim().animator().registerAnimActivity(act);
                //equivalent – metóda vytvárajúca a registrujúca animačnú
//aktivitu
                s.animObject().animateToPositinScaleRotation(
                        new Point2D.Double(100,100),
```

```
                    1.5d, 0d, trvanie);

                …
        }
        …
}
```

PathMoveActivity

Animačná aktivita na premiestnenie animačného objektu z bodu A, v ktorom sa nachádza cez množinu bodov do posledného bodu zadanej množiny so zadaným trvaním celej trasy. Preto že táto aktivita trvá zadaný čas, je ju najlepšie použiť v kontinuálnych asistentoch.

```
public class ProcesPresunu extends ContinualAsistant{
        …
        public void processMessage(MessageForm message) {
                …
                double trvanie = _rnd.next();
                Sprava s = (Sprava) message;
                Point2D[] trasa = new Point2D[]{
                        new Point2D.Double(100,100),
                        new Point2D.Double(100,0)};

                AbstractAActivity act = new PathMoveAActivity(
                        trvanie, s.id(), trasa);

                //alebo
                AbstractAActivity act = new PathMoveAActivity(
                        trvanie, s.id(),
                        new Point2D.Double(100,100),
                        new Point2D.Double(100,0),
                        /*môže nasledovať ľubovolný počet bodov*/);

                mySim().animator().registerAnimActivity(act);

                //equivalent – metóda vytvárajúca a registrujúca animačnú
//aktivitu
                s.animObject().animatePath(
                        trvanie, trasa);
                //alebo
                s.animObject().animatePath(
                        trvanie
                        new Point2D.Double(100,100),
                        new Point2D.Double(100,100),
                        /*môže nasledovať ľubovolný počet bodov*/);
                …
```

```
        }
        …
    }
```

Nastavenie rýchlosti animátora

Rýchlosť animátora sa nastavuje volaním dvoch metód animator().setAnimSpeed(double, double) a animator().setMaxAnimSpeed() na objekte simulácie.

Metóda setMaxAnimSpeed() animátora ABASimu nastaví maximálnu rýchlosť animátora. Ak je aj simulácia nastavená na maximálnu rýchlosť, nebudú delegáti informovaný o potrebe prekreslenia grafického rozhrania.

Pri nastavení rýchlosti animátora ABASimu metódou setAnimSpeed(double, double), berie animátor prvý parameter ako dĺžku intervalu v sekundách a druhý parameter ako trvanie intervalu v sekundách. Ak má animátor ABASimu nastavenú rýchlosť, tak sú informovaný delegáti o potrebe prekreslenia grafického rozhrania.

Ovládanie kanvasu animátora ABASimu

Zaregistrovaný kanvas v grafickom rozhraní zobrazuje stav simulácie. Zobrazovaný stav simulácie sa nemusí zmestiť do vyhradeného priestora pre kanvas, preto kanvas umožňuje posúvanie obrazu a jeho zmenšenie, zväčšenie.

Posúvanie obrazu: držanie ľavého tlačidla myši + posúvanie myšou

Zmenšenie veľkosti obrazu: držanie pravého tlačidla myši + posun myšou doľava

Zväčšenie veľkosti obrazu: držanie pravého tlačidla myši + posun myšou doprava

Súradnicový systém pre animačné objekty v kanvase animátora ABASimu

Súradnicový systém je rovnaký ako pre ostatné Java komponenty. Ak ste neposunuli obrazom, alebo nezmenili jeho veľkosť, bod X:0, Y:0 bude v ľavom hornom rohu. Os Y naberá kladné hodnoty smerom dole. Os X naberá kladné hodnoty smerom doprava.

Potrebné pre animátor ABASimu

Animátor ABASimu využíva na zobrazovanie knižnicu Piccolo2D. Pre použitie animátora ABASimu je nutné pripojiť k projektu knižnicu Piccolo2D.

Diagram tried animátora ABASimu