Дәріс 8. Unity3D-да айнымалылар,функцияларды меңгеру

Мақсаты: Unity3D-да айнымалылар,функцияларды меңгеру
Міндеті: C# де скриптер

Жоспары:
1 iSpring QuizMaker.
2. Интерекпінді тест құрастыру

Базалық класстармен жұмыс
С скриптер

Объектіге бағытталған тілдердің негізгі концепциясы. Объектіге бағытталған тілдердің (Object Pascal, C++, Java, VBasic, SmallTalk және т.б.) негізгі конструкциялары.

Объекті-бағытталған программалау тілдерінің негізгі концепциясы– құрылатын қосымша өзара байланысқан негізгі объектілерден тұрады. Объектібағытталған технологияда қолданушы үш базалық элементпен: объектілер, хабар және класстармен жұмыс істейді. Объектілер дегеніміз бірнеше рет қолданылатын программалық модулдерден, яғни байланысқан мәліметтер мен процедуралардан тұрады. Объект құрылымы екі бөліктен тұрады: айнымалылар және әдістер. Әдістер объект функциясының алгоритмін анықтайтын процедуралар мен функциялар жиынынан тұрады. Объектілі айнымалылар жәй мәліметтерден (сан, массив, текст) және күрделі құрылымды информациялардан (график, дыбыс т.б.) тұрады. Объектілердің өзара байланысуына хабарлар қолданылады және үш бөлімнен тұрады: объект идентификаторы, ағымдағы объектіде қолданылатын әдіс аттары және таңдалған әдіс режимін қалпына келтіретін қосымша информациялар.

Күрделі программалар бірнеше біртипті объектілерді қолдануы мүмкін. Бұл жағдайда әр объект үшін әдістер мен айнымалылар туралы информацияны жазу тиімсіз. Бұл мақсатқа объектілер класы деген түсінік енгізілген. Класс дегеніміз біртипті объектілерге арналған шаблон және объектілі айнымалылар типтері мен әдістерін анықтайтын информациялардан тұрады. Объекті-бағытталған технологияға негізделген программалау тілдері: SmallTalk/v, Object Pascal, ACT++, C++, Simula, Actor, Classic–Aga және т.б. Объекті-бағытталған программалаудың негізгі үш принципі бар: инкапсуляция, тұқымқуалау, полиморфизм.

Ол абстракциялар Жоғарыда біз Миллердің эксперименттеріне сілтеме жасадық, онда әдетте адам бір уақытта тек 7±2 бірлік ақпаратты қабылдай алады. Бұл сан- шамасы, ақпараттың мазмұнына байланысты емес. Миллердің өзі атап өткендей: "Біздің өлшеміміз жад біз алатын ақпарат мөлшеріне қатаң шектеулер қояды қабылдау, өңдеу және есте сақтау. Бір уақытта кіріс ақпаратының түсуін ұйымдастыру бірнеше түрлі арналар арқылы және жеке бөліктер тізбегі ретінде біз бұзу... бұл ақпараттық кептеліс " . Қазіргі терминологияда бұл деп аталады абстракцияларды бөлу немесе бөлектеу

арқылы. Вулф бұл процесті былай сипаттайды: "адамдар өте тиімді технологияны дамытты қиындықты жеңу. Біз одан абстракция жасаймыз.

Толық қалпына келтіре алмау күрделі объект, біз өте маңызды емес бөлшектерді елемейміз және осылайша біз айналысамыз жалпыланған, идеалдандырылған объект моделімен " . Мысалы, фотосинтез процесін зерттеу өсімдіктерде біз белгілі бір жапырақ жасушаларындағы химиялық реакцияларға назар аударамыз біз қалған бөліктерге — шламдарға, тамырларға және т. б. назар аудармаймыз, бірақ біз әлі де олар бір уақытта ақпараттың едәуір мөлшерін қамтуға мәжбүр, бірақ абстракциялар біз мағыналық жағынан едәуір үлкен ақпарат бірліктерін қолданамыз көлемі. Бұл әсіресе әлемді объектіге бағытталған нүктеден қарастырған кезде дұрыс нысандар нақты әлемнің абстракциясы ретінде бөлек болып табылады қанықкан байланысқан ақпараттық бірліктер. 2-тарауда абстракция ұғымы толығырақ қарастырылады.

температуры на C++.

```
// Температура по Фаренгейту
typedef float Temperature;
// Число, однозначно определяющее положение датчика typedef unsigned
int Location;
class TemperatureSensor {
public:
TemperatureSensor (Location);
-TemperatureSensor() ;
void calibrate(Temperature actualTemperature);
Temperature currentTemperature() const;
private:
…
};
```

Мұнда temperature және Location типтерін анықтайтын екі оператор ыңғайлы қарапайым түрлерге арналған бүркеншікним және бұл абстракцияларымызды тілде білдіруге мүмкіндік береді пәндік саласы.

7 Temperature-өзгермелі нүкте пішіміндегі деректердің сандық түрі 7 өкінішке орай, typedef дизайны деректердің жаңа түрін анықтамайды немесе қамтамасыз етпейді оның қорғанысы. Мысалы, келесі сипаттама C в: typedef int Count; Фаренгейт шкаласында температураны жазу үшін. Орналасу түрінің мәндері ферма орындарын білдіреді температура датчиктері орналасуы мүмкін.

Мысалы, сіз осылай жаза аласыз:

```
Temperature temperature;
TemperatureSensor greenhouselSensor(l);
TemperatureSensor greenhouse2Sensor(2) ;
temperature = greenhouselSensor.currentTemperature();
```

Ағымдағы температура операциясына қатысты инварианттарды қарастырыңыз. Алғышарт сенсор жылыжайда дұрыс жерде орнатылған деген болжамды қамтиды, ал кейінгі шарт — бұл сенсор температура мәнін

Фаренгейт градусымен қайтарады. Осы уақытқа дейін біз сенсорды пассивті деп санадық: біреу одан температураны сұрауы керек және сонда ол жауап береді. Дегенмен, тағы бір, бірдей жарамды тәсіл бар. Сенсор белсенді болуы мүмкін температураны бақылаңыз және оның берілген мәннен ауытқуы болған кезде басқа объектілерге хабарлаңыз берілген деңгейден асады. Осыдан Абстракция аз өзгереді: сәл басқаша объектінің жауапкершілігі тұжырымдалады. Осыған байланысты оған қандай жаңа операциялар қажет? Мұндай жағдайларда жиі кездесетін идиома-кері қоңырау. Клиент серверге ұсынады функция (кері қоңырау шалу функциясы) және сервер оны қажет деп санаған кезде шақырады. Мұнда қажет сияқты нәрсе жазыңыз:

```
class ActiveTemperatureSensor {
public:
ActiveTemperatureSensor (Location,
void (*f)(Location, Temperature));
~ActiveTemperatureSensor() ;
void calibrate(Temperature actualTemperature);
void establishSetpoint(Temperature setpoint,
Temperature delta);
Temperature currentTemperature() const;
private:
…
};
```

ActiveTemperatureSensor жаңа класы сәл күрделене түсті, бірақ жеткілікті жаңа абстракцияны білдіреді. Сенсордың данасын жасай отырып, біз оны инициализациялау кезінде береміз орын ғана емес, сонымен қатар параметрлері анықтайтын кері қоңырау функциясының көрсеткіші орнату орны мен температурасы.

Жаңа establishsetpoint орнату мүмкіндігі мүмкіндік береді клиентке температура сенсорының іске қосылу шегін өзгерту керек, ал сенсордың жауапкершілігі ағымдағы температура болған сайын кері қоңырау функциясын шақыру actualTemperature Delta-ға қарағанда setpoint-тен ауытқиды. Бұл ретте клиентке іске қосу орны мен ондағы температура белгілі болады, содан кейін ол өзі білуі керек, бұл туралы не істеу керек. Тұтынушы әлі де температураны өз бетінше сұрай алатынын ескеріңіз бастама. Бірақ егер клиент инициализация жасамаса, мысалы, рұқсат етілмесе ше температура?

Жобалау кезінде біз бұл мәселені міндетті түрде шешуіміз керек- ақылға қонымды болжам бар ма: температураның рұқсат етілген өзгеру аралығы деп есептелсін шексіз кең. Activetemperaturesensor класы өз міндеттемелерін қалай орындайтынына байланысты оның ішкі көрінісі және сыртқы клиенттерді қызықтырмауы керек. Бұл анықталады оның жабық бөлігі мен мүше функцияларын іске асыру арқылы жүзеге асырылады. қарапайым Int түріне синонимді енгізеді. Келесі бөлімде көретініміздей, басқа тілдер, Ada және Eiffel сияқты негізгі типтерді қатаң теруге қатысты жетілдірілген семантикаға ие. C++ - да өсіру жоспары келесідей болады. Алдымен жаңаларын енгізейік

деректер түрлері, біздің абстракцияларымызды Домен сөздігіне жақындату (күн, сағат, жарықтандыру, қышқылдық, концентрация):

```
// Число, обозначающее день года
typedef unsigned int Day;
// Число, обозначающее час дня
typedef unsigned int Hour;
// Булевский тип
enum Lights {OFF, ON};
// Число, обозначающее показатель кислотности в диапазоне от 1 до 14
typedef float pH;
// Число, обозначающее концентрацию в процентах: от 0 до 100
typedef float Concentration;
```

Далее, в тактических целях, опишем следующую структуру:

```
// Структура, определяющая условия в теплице
struct Condition {
Temperature temperature;
Lights lighting;
pH acidity;
Concentration concentration;
};
class GrowingPlan (
public:
GrowingPlan (char *name);
virtual
~GrowingPlan();
void clear();
virtual void establish(Day, Hour, const Condition&);
const char* name() const;
const Condition& desiredConditions(Day, Hour) const;
protected:
…
};
```

Біз бір жаңа жауапкершілікті қарастырғанымызды ескеріңіз: әр жоспардың аты бар және ол орнатуға және сұрауға болады. Сонымен қатар, establish операциясы келесідей сипатталғанын ескеріңіз ішкі сыныптар оны қайта анықтай алатындай виртуалды.

Сипаттаманың ашық (қоғамдық) бөлігінде объектінің конструкторы мен деструкторы көрсетілген (оны құру және жою процедураларын анықтайтын), модификацияның екі процедурасы (бүкіл clear жоспарын тазарту және establish жоспарының элементтерін анықтау) және екі анықтаушы селектор күйлер (Name және desiredcondition функциялары). Біз сипаттамада жабық бөлікті қалдырдық оны эллипсиспен алмастыратын сынып, өйткені біз қазір сыртқы жауапкершілікті емес, маңыздымыз сыныптың ішкі көрінісі. Инкапсуляция Бұл нені білдіреді? Біз GrowingPlan абстракциясын салыстыру

ретінде сипаттағанымызбен уақыт сәттеріне әрекет ету, оны кесте ретінде жүзеге асырудың қажеті жоқ деректер.

Шынында да, клиентке оның сыныбын жүзеге асыруға ешқандай себеп жоқ ол өз міндеттемелерін орындағанға дейін қызмет етеді. Шын мәнінде, абстракция нысан әрқашан оны іске асырудан бұрын болады. Ал іске асыру туралы шешім қабылданғаннан кейін, ол көптеген клиенттерден жасырылған абстракцияның құпиясы ретінде түсіндірілуі керек. Қалай ақылды Ингалс: "күрделі жүйенің ешбір бөлігі ішкі құрылғыға тәуелді болмауы керек кез келген басқа бөлік" . Ал абстракция " адамдарға олар туралы ойлауға көмектеседі "инкапсуляция "бағдарламаларды оңай қайта құруға мүмкіндік береді" .

Абстракция мен инкапсуляция бірін-бірі толықтырады: абстракция мыналарға бағытталған нысанның бақыланатын әрекеті, ал инкапсуляция ішкі құрылыммен айналысады. Көбінесе инкапсуляция ақпаратты жасыру арқылы, яғни барлығын жасыру арқылы жүзеге асырылады сыртқы мінез-құлыққа әсер етпейтін ішкі бөлшектер. Әдетте ішкі жағы да жасырылады объектінің құрылымы және оның әдістерін жүзеге асыру.

Инкапсуляция осылайша әр түрлі арасындағы нақты шекараларды анықтайды абстракциялармен. Мысал ретінде өсімдік құрылымын алайық: жоғарғы деңгейде түсіну үшін Фотосинтездің әрекеті, тамыр функциялары сияқты егжей-тегжейлерді елемеуге болады өсімдіктер немесе жасуша қабырғаларының химиясы. Сол сияқты мәліметтер базасын жобалау кезінде бағдарламаларды деректердің физикалық көрінісіне тәуелді болмайтындай етіп жазу әдеттегідей; оның орнына деректердің логикалық құрылымын көрсететін схемаға назар аударыңыз.

Интерфейсті бөлу принципі және іске асыру заттардың мәніне сәйкес келеді: интерфейсте барлық нәрсе бар осы объектінің кез келген басқа объектілермен өзара іс-қимылы; іске асыру басқалардан жасырады объектілердің өзара әрекеттесу процесіне қатысы жоқ барлық бөлшектер. Британ және Парнас мұндай мәліметтерді "абстракция құпиялары" деп атады .

Бақылау сұрақтары:

1. С скриптер дегеніміз не?

2. SmallTalk/v, Object Pascal, ACT++, C++, Simula, Actor, Classic–Aga программалау тілдері