Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и информатики

Кафедра прикладной математики и кибернетики

КУРСОВАЯ РАБОТА

по предмету объектно-ориентированное программирование

«Виртуальный мир»

Выполнил: студент группы ИП-312

Бочкарев Б.В.

Проверил:

Климова И.В.

Новосибирск 2014

**Содержание**

|  |  |
| --- | --- |
| 1. Постановка задачи------------------------------------------------------------------------- | стр 3 |
| 1. Основные идеи применяемых методов----------------------------------------------- | стр 4 |
| 1. Классы в программе----------------------------------------------------------------------- | стр 5 |
| 1. Вывод---------------------------------------------------------------------------------------- | стр 9 |

1. **Постановка задачи.**

С помощью ООП разработать возможную реализацию некоторого компьютерного мира. Обитатели такого мира могут иметь различную форму, быть подвижными и неподвижными, быстрыми и медленными, могут размножаться, скрещиваться, нападать и защищаться и т.д. (выполнять могут до трех человек в группе).

**Необходимый минимум содержания работы:**

* Инкапсуляция (все поля данных не доступны из внешних функций)
* Наследование (минимум 3 класса, один из которых - абстрактный)
* Полиморфизм
* Конструкторы, Перегрузка конструкторов
* Желательно: минимум + 2 технологии ООП (статические элементы, дружественные функции, классы, виртуальные функции, шаблоны, множественное наследование, массивы указателей на объекты, конструкторы копирования, параметры по умолчанию, использование объектов в качестве аргументов или возвращаемых значений).

1. **Основные идеи и характеристики применяемых методов.**

**Объектно-ориентированное программирование (ООП)** — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя. Во всех классах программы реализовано разделение на protected и public блоки. Работать с полями из protected блока могут только методы из public и методы из public наследников.

Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. При использовании термина «полиморфизм» в сообществе ООП подразумевается полиморфизм подтипов; а использование параметрического полиморфизма называют обобщённым программированием.

Класс является описываемой на языке терминологии исходного кода моделью ещё не существующей сущности (объекта). Фактически он описывает устройство объекта, являясь своего рода чертежом. Говорят, что объект — это экземпляр класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

Объект - сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

Абстрактный класс — базовый класс, который не предполагает создания экземпляров. Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором объявлен, однако должен быть реализован для его неабстрактных потомков. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объём и наименьшее содержание.

Виртуальный метод — метод класса, который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения. Таким образом, программисту необязательно знать точный тип объекта для работы с ним через виртуальные методы: достаточно лишь знать, что объект принадлежит классу или наследнику класса, в котором метод объявлен.

Программа для курсовой работы написана с использованием OpenGL.

1. **Классы программы**

class Atom{

public:

virtual void Draw() = 0;

}; // абстрактный класс с чистым виртуальным методом, используется, как прародитель всех остальных классов программы

class People:public Atom{

private:

int id; // идентификатор жителя

int statusGo; // в какую сторону движется

int sex; // 0 - мужчина, 1 - женщина

int x, y; // координаты людей

int idMarriage; // "семья"

int idWar; // "враг"

int health; // здоровье

int children; // количество детей

int speed; // скорость

Home myHome; // дом персонажа

public:

int GetSex(){ return sex; }

int Getx(){ return x; }

int Gety(){ return y; }

int GetIdMarriage(){ return idMarriage; }

int GetIdWar(){ return idWar; }

int GetHealth(){ return health; }

int GetChildren(){ return children; }

void SetIdMarriage(int id){ idMarriage = id; }

void SetIdWar(int id){ idWar = id; }

void SetChildren(){ children++; }

Home GetHome(){ return myHome; }

People(){}

People(int idM, int X = 0, int Y = 0){

speed = rand()%3+1;

health = rand()%100+50;

//printf("%d\n",health);

children = 0;

id = idM;

idWar = idMarriage = -1;

if(idM == 0) sex = 0;

else if(idM == 1) sex = 1;

else sex = rand()%2;

x = X;

y = Y;

statusGo = 0;

}

void Live(){ // главный цикл жизни

Draw();

Dvig();

if(rand()%10 == 3){

Merriage(id,sex);

}

if(rand()%50 == 5 && idMarriage != -1){

NewClass(id,idMarriage);

}

if(rand()%500 == 8){

War(id);

}

if(rand()%300 == 12 && this->myHome.Status() != 1){

if(NewHome(this->myHome, this->x, this->y, this->id) == 1){

this->myHome.Init(this->x,this->y,(float)rand()/10000,(float)rand()/100000,(float)rand()/100000);

printf("%d построил дом!\n",this->id);

}

}

}

void Draw(){ // отрисовка персонажей

int tempX = x, tempY = y;

if(sex == 1){

glBegin(GL\_TRIANGLES);

glColor3f(1.0,1.0,1.0);

glVertex2f(tempX, tempY + 25/2);

glColor3f(0.0,0.0,1.0);

glVertex2f(tempX + 25/2, tempY-25/2);

glColor3f(1.0,0.0,1.0);

glVertex2f(tempX - 25/2, tempY-25/2);

glEnd();

} else if(sex == 0){

glBegin(GL\_TRIANGLES);

glColor3f(0.0,0.0,1.0);

glVertex2f(tempX-25/2, tempY+25/2);

glColor3f(1.0,0.0,0.0);

glVertex2f(tempX+25/2, tempY+25/2);

glColor3f(1.0,1.0,0.0);

glVertex2f(tempX, tempY-25/2);

glEnd();

}

if(this->myHome.Status() == 1){

this->myHome.Draw();

}

}

void Dvig(){ // движение персонажа

int go = rand()%100;

switch(go){

case 12: statusGo = 0; break;

case 8: statusGo = 1; break;

case 10: statusGo = 2; break;

case 3: statusGo = 3; break;

}

switch(statusGo){

case 0: if(x+25/2<WinWid/2) x+=speed; break;

case 1: if(x-25/2>-WinWid/2) x-=speed; break;

case 2: if(y+25/2<WinHei/2) y+=speed; break;

case 3: if(y-25/2>-WinHei/2) y-=speed; break;

}

}

}; // класс служит для создания объектов – людей, за их перемещение и «жизнь», в которой все определяется случаем.

class Town:public Atom{

protected:

int status; // активность

int x, y; // координаты Объектов

public:

Town(){}

Town(int x, int y){

this->x = x;

this->y = y;

}

int Status(){ return this->status; }

int GetX(){ return this->x; }

int GetY(){ return this->y; }

}; // класс-родитель для таких объектов, из которых может состоять город, имеет только самые основные методы и поля

class Home:public Town{

protected:

float color1; // цвет дома

float color2; // цвет дома

float color3; // цвет дома

public:

Home(){}

void Init(int x, int y, float color1, float color2, float color3){

this->x = x;

this->y = y;

this->status = 1;

this->color1 = color1;

this->color2 = color2;

this->color3 = color3;

}

void Draw(){

glBegin(GL\_QUADS);

glColor3f(this->color1,this->color2,this->color3);

glVertex2f(this->x, this->y);

glVertex2f(this->x + 25, this->y);

glVertex2f(this->x+25, this->y+25);

glVertex2f(this->x, this->y+25);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2f(this->x, this->y + 25);

glVertex2f(this->x + 25/2, this->y + 25 + 25/2);

glVertex2f(this->x + 25, this->y + 25);

glEnd();

glBegin(GL\_LINES);// домик

glColor3f(1.0,1.0,1.0);

glVertex2f(this->x+25/4,this->y+25-25/4); // окошко

glVertex2f(this->x+3\*(25/4),this->y+25-25/4);

glVertex2f(this->x+3\*(25/4),this->y+25-25/4);

glVertex2f(this->x+3\*(25/4),this->y+25/4);

glVertex2f(this->x+3\*(25/4),this->y+25/4);

glVertex2f(this->x+25/4,this->y+25/4);

glVertex2f(this->x+25/4,this->y+25/4);

glVertex2f(this->x+25/4,this->y+25-25/4);

glVertex2f(this->x+2\*25/4,this->y+25/4);

glVertex2f(this->x+2\*25/4,this->y+25-25/4);

glVertex2f(this->x+25/4,this->y+2\*(25/4));

glVertex2f(this->x+2\*25/4,this->y+2\*(25/4));

glEnd();

}

}; // класс для создания домов на игровом поле, содержит в себе прорисовку и создание

class Picea:public Town{

public:

Picea(){}

Picea(int x, int y):Town(x,y){};

void Draw(){

glColor3f(0.0,1.0,0.0);

glBegin(GL\_TRIANGLES);

glVertex2f(this->x-12, this->y);

glVertex2f(this->x+12, this->y);

glVertex2f(this->x, this->y+12);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2f(this->x-12/1.25, this->y+12);

glVertex2f(this->x+12/1.25, this->y+12);

glVertex2f(this->x, this->y+12+12/1.25);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2f(this->x-12/2, this->y+12/1.25);

glVertex2f(this->x+12/2, this->y+12/1.25);

glVertex2f(this->x, this->y+12+12/1.25+12/2);

glEnd();

glBegin(GL\_QUADS);

glColor3b(150,75,0);

glVertex2f(this->x-12/4, this->y);

glVertex2f(this->x+12/4, this->y);

glVertex2f(this->x+12/4, this->y-12/4);

glVertex2f(this->x-12/4, this->y-12/4);

glEnd();

}

}; // класс для озеленения города, рисует «ели», имеет конструктор, который отправляет данные родителю для заполнения полей

1. **Вывод**

В курсовой работе реализован виртуальный мир с использованием OpenGL. Использовались такие методы ООП, как наследование, полиморфизм, абстрактные классы с чистыми виртуальными методами, передача объекта в функцию, объект класса является полем другого объекта.