

Муниципальная гимназия №30 им. Д.Н. Музалева

# Компьютерное моделирование физических эффектов и явлений

*Выполнил: Антипов Дмитрий, 11-Ц класс.*

*Руководитель: Нестеренко А.А.,  
учитель информатики и ТРИЗ.*

Петрозаводск  
2002

Оглавление

ВВЕДЕНИЕ ..... 2

	1
<b>ЭТАПЫ РАБОТЫ .....</b>	<b>2</b>
1-й ЭТАП РАБОТЫ – МОДЕЛИРОВАНИЕ ПО ММЧ .....	2
<i>Метод Маленьких Человечков (ММЧ)</i> .....	2
2-й ЭТАП – СОЗДАНИЕ КОНСТРУКТОРА ФИЗИЧЕСКИХ ЯВЛЕНИЙ .....	3
<b>ВЫБОР СРЕДСТВ ПРОГРАММИРОВАНИЯ .....</b>	<b>3</b>
<b>ОБЪЕКТНЫЙ ПОДХОД В ПРОГРАММИРОВАНИИ .....</b>	<b>4</b>
<b>ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ.....</b>	<b>4</b>
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>5</b>
<b>СПИСОК ЛИТЕРАТУРЫ.....</b>	<b>5</b>
<b>ПРИЛОЖЕНИЕ 1. ОБЪЕКТНАЯ СТРУКТУРА ПРОГРАММЫ.....</b>	<b>6</b>
<b>ПРИЛОЖЕНИЕ 2. ФРАГМЕНТЫ ПРОГРАММЫ С ОПИСАНИЕМ ОБЪЕКТОВ.....</b>	<b>7</b>

## Введение

Физические эффекты и явления – важная часть информационного фонда ТРИЗ. Фонд физических эффектов очень велик, и изобретателю нужно в нем ориентироваться. Запомнить все эффекты невозможно, да и не нужно, но наиболее часто употребляемые эффекты стоит держать в уме. Сделать это гораздо легче, если эффект запоминается в виде картинки.

Цель данной работы - создать программу, представляющую собой компьютерную модель физических явлений и эффектов. Зачастую бывает сложно адекватно изобразить процесс так, чтобы чётко отражалась физическая сторона явления, и при этом любому человеку было бы понятна сущность происходящего. Компьютер же, я думаю, расширяет возможности по моделированию, позволяет изображать более сложные процессы, не теряя при этом наглядности.

Хорошо созданный образ может помочь в понимании и запоминании процесса. Поэтому моя главная задача – это создать наглядный и точный образ.

## Этапы работы

Данную работу можно разбить на 2 этапа.

### 1-й этап работы – моделирование по ММЧ

На первом этапе я просто создавал модели физических явлений. Это были различные тепловые явления (изменение температуры, объёма, агрегатного состояния вещества), волновые процессы, а также такие, как диффузия, броуновское движение и другие. Причём для создания всех этих моделей, я использовал метод маленьких человечков (ММЧ).

Метод Маленьких Человечков был предложен Г.С. Альтшуллером для моделирования процессов, происходящих в оперативной зоне, для решения изобретательской задачи. Позднее группа преподавателей ТРИЗ (Б.Л. Злотин, М.Н. Шустерман и др. модифицировали этот метод для обучения).

### Метод Маленьких Человечков (ММЧ)

Кратко опишу этот, многим известный, метод. Главная идея состоит в том, что все частицы (атомы, молекулы) изображаются в виде человечков. Все человечки разделяются на три типа: человечки твёрдого тела, жидкости и газа. Твёрдое тело изображается в виде человечков, стоящих близко друг к другу и держащихся за руки (рис.1). Жидкость изображается в виде человечков, также близко стоящих друг к другу, но за руки они не держатся (рис.2). Газ изображается в виде далеко стоящих друг от друга человечков (рис.3). Этот метод позволяет ярко представить моделируемое явление. Особенно хорошо этот метод воспринимается детьми младших классов.

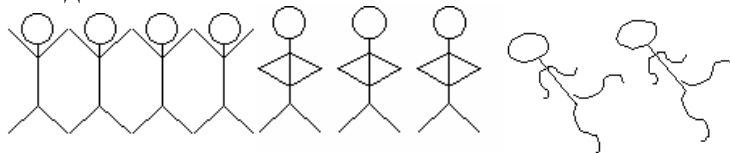


Рис.1

Рис.2

Рис.3

Мною написаны 2 пакета программ:

1). Пакет для компьютера «Макинтош», (система ЛОГО) Мир содержащий программы:

- движение «человечков»;
- тепловое расширение;
- звук;
- эхо;
- противогаз (процесс фильтрации);
- конденсация;
- испарение;
- кипение;
- диффузия.

2). Пакет для IBM на языке Java Script, содержащий программы:

- |               |                         |
|---------------|-------------------------|
| - движение;   | - броуновское движение; |
| - звук;       | - диффузия;             |
| - деформации; | - тепловое расширение.  |

Эти программы выложены на нашем сайте в Интернете ([home.onego.ru/~alla\\_triz](http://home.onego.ru/~alla_triz)).

На этом этапе программировались, в основном, молекулярные явления, т.к. модель ММЧ «работает» только на молекулярном уровне.

## 2-й этап – создание конструктора физических явлений

Целью второго этапа работы является создание конструктора физических явлений, позволяющего пользователю смоделировать любой физический эффект по его желанию. Здесь возникает проблема, которая заключается в том, что довольно трудно поместить весь перечень физических эффектов в какую-то одну схему. Я попытался решить эту проблему и создал наиболее подходящую структуру. На входе у пользователя запрашивается необходимая информация о физическом эффекте, который он хочет получить. Во-первых, запрашивается информация о самом объекте, посредством которого осуществляется эффект. Это может быть: вид вещества, внутренние характеристики вещества, такие, как температура, плотность и т.д. Во-вторых, запрашивается информация о внешнем воздействии на этот объект, т.е. природа поля (МАТХЭМ), сила поля и другие. И, возможно, запрашивается результат, который нужно получить. В результате работы программы согласно введенным данным будет осуществляться моделирование эффекта. Данная структура ввода информации является универсальной, т.е. с ее помощью можно моделировать довольно много различных физических эффектов. Вместе с тем, возникает проблема программирования. Довольно трудно создать программу, которая включает в себя такое разнообразие возможностей.

В настоящее время я занимаюсь написанием программ, моделирующих электромагнитные эффекты. Именно эти эффекты первыми будут включены в конструктор.

## Выбор средств программирования

В качестве средства программирования я выбрал язык «Pascal». Это выбор был сделан по двум причинам, во-первых, он содержит основные графические средства, которые необходимы мне для непосредственного изображения моделей. И, во-вторых, в этом языке реализован объектный подход. Именно с помощью объектного подхода я собираюсь описать как можно больше физических эффектов и связать их в единую структуру, которой будет легче управлять. Так же при создании системы классов, достигается более лёгкое дополнение программы. А это очень важно при продолжении работы.

## Объектный подход в программировании

Любое программирование, - это выполнение операций над данными. Но описывать этот процесс можно по-разному. Рассмотрим в сравнении два подхода: процедурный и объектно-ориентированный.

*Процедурный подход.* Операции, они же процедуры, они же программные модули, существуют независимо от данных. И существует некий Исполнитель способный выполнять процедуры. Он берёт очередную процедуру, берёт данные, которые эта процедура требует, и выполняет одну за другой все инструкции, описанные в процедуре. Дело же программиста определить Исполнителю порядок исполнения процедур и описать содержимое каждой процедуры.

*Объектно-ориентированный (ОО) - подход.* Процедуры и данные, обрабатываемые этими процедурами, объединены в одну структуру, называемую классом. Данные класса можно обрабатывать только процедурами класса, а процедуры класса можно использовать только с дан-

ными класса.

В действительности нет никакой необходимости делать процедуры способными обрабатывать любые данные. Процедура, всегда кроме текста на языке программирования несёт в себе определённый смысл, закладываемый в неё программистом. И этот смысл требует и вполне определённых данных. То есть связь между данными и процедурой существует объективно. Технология объектно-ориентированного программирования всего лишь требует указать эту связь явным образом.

Программа, это тоже класс, состоящий из всех свойств и всех методов, но этот большой класс можно разбить на независимые подклассы, те в свою очередь на другие подклассы и так до тех пор, пока не дойдём до элементарных классов. Таким образом, программы можно собирать из представителей классов как из кирпичиков. В структурном программировании это тоже было известно. Такой метод называется декомпозицией. Но в ОО - программировании декомпозиция значительно глубже, так как строительные кирпичики представляют собой не просто процедуры, а процедуры с данными. Эти кирпичики полновеснее.

При объектном подходе гораздо легче использовать уже имеющиеся наработки. Готовый объект предоставляет сразу целый комплекс услуг по обработке нужных данных, в то время как готовые библиотечные процедуры не заботятся о том, как будут обрабатываться результаты их деятельности другими процедурами.

Класс, как уже говорилось выше, можно строить из классов. Что это означает? Как именно построить большой класс из маленьких? Ответ на вопрос даёт принцип наследования. Этот принцип гласит, что для двух классов можно установить родственные отношения. А именно один из них объявить родительским, а второй дочерним. Если это сделано, то все свойства и методы родительского класса становятся известными дочернему классу. А так как любой класс может быть для одного класса дочерним, а для другого родительским, то появляется возможность строить очень длинные родственные цепочки. Можно даже построить целое дерево классов. В ОО - программировании построение дерева классов это и есть разработка программы.

Коротко можно сказать так: принцип наследования - это механизм передачи свойств и методов.

Что такое объекты и чем они отличаются от классов? Чтобы ответить на этот вопрос, вспомним, чем отличается тип данных от самих данных. То есть чем отличаются две записи: `type r=integer` и `var r:integer`.

В первой записи переменной с указанным именем не существует, это название типа и чтобы появилась переменная, я должен сделать ещё описание переменной `var u:r`; во втором же случае мы сразу имеем дело с реальной переменной для которой отведена реальная физическая область памяти. Так же и класс - это тип переменной, которую ещё надо объявить, а называется эта переменная объектом. Другими словами, объект - это реально существующая сущность в то время, как класс, это абстрактная сущность, не существующая в реальности.

Сказанное выше означает, что объект создается в процессе работы программы, существует какое-то время, выполняя свои функции, и может быть уничтожен, в то время, как класс описывается на этапе разработки, но не создается и не уничтожается в процессе выполнения программы.

## Структура программы и данных

Действительно, если рассматривать объектный подход в применении к описанной выше задаче, то можно обнаружить существенные преимущества в использовании объектной структуры. Система классов способна привести в порядок даже очень большой перечень эффектов. При разработке моделей это позволяет идеализировать программы. Во многом объектный подход помогает самому процессу программирования. В больших программах довольно трудно удержать в голове всю систему. С ростом количества процедур, данных воз-

можно потерять логическую цепь выполнения команд. Но при чёткой структуре, раскладывающей по полочкам все данные и методы их обработки, понять работу программы становится проще. Так же в этом случае облегчается дополнение и изменение некоторых частей программы.

В настоящее время составлена система классов для конструктора, включающего некоторые электромагнитные физические эффекты (см. схему в приложении 1). Эта схема лежит в основе программы.

## Описание работы программы

Итак, теперь кратко опишу алгоритм работы конструктора. В самом начале пользователю предлагается ознакомиться с правилами введения данных, а так же с теоретической стороной понятия «физический эффект». Затем на экран выводится меню, которое позволяет выбирать ресурс эффекта и поле воздействия на него, а также выполнять некоторые непосредственные команды.

На основе введённых данных выбирается класс и методы. И далее создаётся необходимый объект, моделирующий эффект. После его демонстрации программа вновь возвращается к меню для дальнейших указаний.

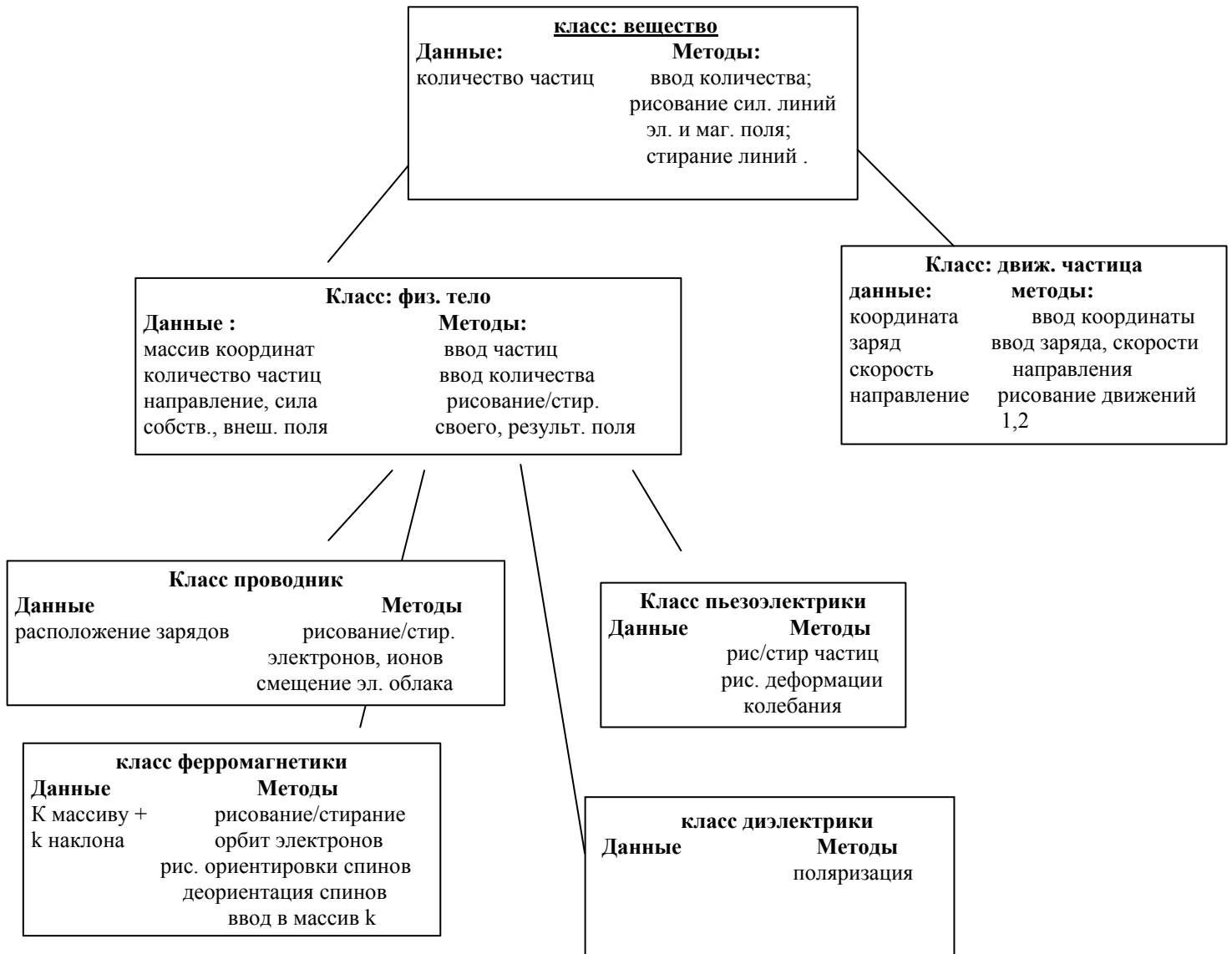
## Заключение

На данный момент запрограммирована система классов электромагнитных явлений, а также сами модели разных типов физических явлений. Предстоит дополнить систему классов новыми представителями, включить в нее классы, с помощью которых моделируются эффекты другого рода. Но основную часть работы можно считать уже выполненной, поскольку система работы конструктора уже разработана и приведена в рабочее состояние.

## Список литературы

1. Гради Буч. Объектно-ориентированное проектирование с примерами применения.-М.: Конкорд, 1992.
2. Б.Л. Злотин, А.В. Зусман. Изобретатель пришел на урок.-Кишинев: Лумина, 1988
3. Дерзкие формулы творчества./ Сост. А.Б. Селюцкий. - Петрозаводск: Карелия, 1987. - (Техника - молодежь - творчество).
4. Дерк Луис. Справочник. С и С++.-М.: Бином, 1997.
5. Иванов Г.И. Формулы творчества, или как научиться изобретать: Книга для учащихся старших классов. - М.: Просвещение, 1994.
6. А. Федоров. Java Script для всех.-М.: Компьютер пресс, 1998.

# Приложение 1. Объектная структура программы



## Приложение 2. Фрагменты программы с описанием объектов

```

type vesh=object {объект вещество}
br:boolean; {параметр поля}
procedure el; {рисование эл полей}
procedure el_s1; {рисование своего поля}
procedure el_s2;
procedure mag; {рисование маг поле}
procedure mex; {рисование возрастания температуры}
procedure dark; {очистка экрана}
procedure ss1; {звук1}
procedure ss2; {звук2}
end;

```

```

type electron=object(vesh) {объект частица}
v,x,y,k:real;{скорость координаты направление}
constructor nn(fl:byte); {конструктор}
destructor dd; {деструктор}
procedure nach; {установление v,x,y,k}
procedure dv; {простое движение}
procedure dv_el; {движение с эл полем}
procedure dv_mag; {движение с маг полем}
end;

```

```

type telo=object(vesh) {объект тело}
m:array [1..5,1..100] of real; {массив координат}
procedure draw; {рисование тела1}
procedure draw_el;{рисование тела2}
end;

```

```

type segn=object(telo)
constructor nn(fl,f:byte);
destructor dd;
procedure eff;
procedure termo;
end;

```

```

type dial=object(telo) {объект диэлектрик}
procedure eff1; {действие поля}
constructor nn(fl:byte); {конструктор}
destructor dd; {деструктор}
end;

```

```

type prov=object(telo) {объект проводник}
br:boolean; {индикатор поля}
procedure eff2; {воздействие поля}
constructor nn(fl:byte); {конструктор}
destructor dd; {деструктор}
procedure reshot; {рисование решётки}
procedure draw1; {рисование электронов}
end;

```