CHAPTER : IV
COMPUTER GRAPHICS
&
COMPUTER AIDED LEARNING

CHAPTER- IV :: COMPUTER GRAPHICS AND COMPUTER AIDED LEARNING ::

4.1 INTRODUCTION

The following Chinese proverb sums up a pedagogical philosophy for teachers and software developers to keep in mind when they work with young learners.

I HEAR, I FORGET.

I READ, I UNDERSTAND.

I DO, I REMEMBR.

So it seems that training is essential to learn perfectly. But modern world is the world of high technology and there are many areas where training is prohibitively expensive. e.g. to train pilots for civil and military aircrafts, for ships, submarines and space missions, require large resources of time, money and equipment. To model products or processes in computer aided design, Chemistry, oil-drilling, medicine, engineering, power station control demands equally expensive resources. Computer aided learning (CAL) is an effective solution on all the above difficulties. CAL involves simulting part of the real word and giving interactive control of that simulation to an operator. With support from researchers and teachers in designing appropriate instruction tools, difficult science topics and experiments may become more inspiring and more interactive to most students in school. Moreover it makes learning an enjoyable experience. Here we must keep in mind

that computer simulation or CAL is not a total substitute for experiments, neither at the tertiary level nor at the secondary level. It is just an useful tool in pre-training course which would save the expensive resources.

4.2 COMPUTER AIDED DRAWING IN COMPUTER AIDED LEARNING

Computer was applied in teaching since the early days of invention. It effectively teaches concepts and practical skills in a very short time. Use of computer graphics in CAL programs adds a new dimension and makes it exciting and dynamic. In the old days CAL was only limited to scientific studies e.g. learning diffraction in chemistry or calculus in mathematics but now PC's are common everywhere and the range of application of CAL is also increasing. Computer aided drawing (or computer graphics) has a major role in todays CAL applications. This can be better proved by taking the example of CAL program developed by Robin Hight[7] at MJohns Hopkins University. This program helps people with defective hearing master to learn the difficult art of lip-reading. The program converts the typed sentence into a corresponding sequence of lip shapes. There are in all 19 distinct phonetic sound. One can see the animated sequence of lips, tongue and teeth on the screen. The entire system is implemented on APPLE II microcomputer. The memory available with APPLE II is 48 KB and actually this is an insufficient memory to handle such a big graphic task. If the program would have been written on pixel by pixel definition of the shapes of lips, tongue and teeth for 19 distinct phonetic sounds then the memory is clearly insufficient for storing the data and to handle the disc operating

system (DOS). An intelligent solution to this problem is to break down the program into a library of components which forms the basic primitives. For each phoneme the basic primitives are called with different orientation. So the program 'Lip-reader' is the ideal illustration of how computer aided drawing can be used on any size of computer to teach difficult skills or concepts.

4.3 SIMULATION

Certainly simulation has a major place in computer aided learning,[15] Simulation is nothing but running the process in software. It is generally considered as a process of developing a model of the desired system and experimenting with it to gain an insight into the interesting aspects of the system. Simulation methodology is based on computer science, statistics, and operation research and is now sufficiently developed and coherent to be called as a discipline in its own right. A course in simulation is essential in any computer science program. Simulation tool is widely applied in engineering, business and in physical and social sciences.

Why are we interested at all to work with a model of the system rather than the actual system ? Following reasons will answer this question

a) Working with the system may be too expensive.

It is desired, to study sailing of a ship through known

and unknown water, and testing the ability of a navigator to react to a series of situations. If would be clearly, too expensive to actually smash a ship at each time. Here we can use a realistic simulation system constructed by Marconi.[7] The system is known as TEPIGEN (Television Picture Generator).

b) Working with the system may be too dangerous

In the fields like aviation, it is clearly unacceptable to experiment with human beings. In such situation flight simulator systems can be used. CT-5[7] is a very advanced Anglo-American System and provides significantly higher levels of details, texture and flexibility. It is not only limited to military training but also to the ship handling and engineering development. Helmet-mounted laser projection systems have also been developed in which the display performance, interms of field of view and resolution are matched to the visual performance of the pilot's eye.

c) Working with the system may be impossible

Jupitor-Shoemaker collision is the recent hot event. In such events it is impossible to actually work with the systems and collect the data. Computer simulation had shown the effects of the collision before hand.

d) Black Box problem

National and world economies, and other problems like weather, are poorely understood real systems. Modelling helps in understanding these problems.

4.3.1 TYPES OF SIMULATION :

There are three broad classes of simulation :

i) Analog simulation .

ii) Digital simulation.

iii) Hybrid simulation.

i) Analog simulation

Here the problem is formulated in the form of a differential equation.and normally solved by an analogue machine Analog computer is a machine implemented with operational amplifiers and difference amplifiers. The problems like-wind tunnel test on scaled down airplane wings, heat transfer problems, thermodynamic and fluid dynamic problems can be effectively solved with the help of analog simulation.

ii) Digital simulation

Here digital computers are used to solve the differential equations that models the system to be simulated. Discrete event simulation problems are often solved by digital simulation.

iii) Hybrid simulation

A hybrid computer system is a system with high speed CPU and it augments parallal analog processor. Launder (1976) showed that the computing speed of such a computer is considerably higher than a pure digital or pure analog system.

4.3.2 SIMULATION AND MODELLING

Though simulation includes modelling, there are certain points that distinguish between them. Simulation allows the

learner to control input parameters and observe the output, while modelling allows the student to control or change the model on which the simulation is based. It is found that, changing the model increases the understanding of the laws of the system. A program called 'MOTION' is developed by Bork (quoted in reference 12), provides the student to change the inverse seqare law of gravitational attraction which would result in unstable gravitational orbits. This result in a good understanding of the law of gravitational attraction.

Modelling a real system is really ad hoc. A typical example of a real system simulation and its model is shown in figure number 3 . So it is said that "In practical simulation studies, there is usually no interest in steady state behaviour; start up and end effects do form part of the relevant output" Despite this, model based on steady state approximation are very useful.

So computer simulation can be divided into following Steps :

- Study the system,
- Develop a model,
- Develop a computer program that represents the model.
- Design meaningful experiments for the model.
- Carry out simulation runs.
- Interpret the results in terms of properties of the original system.

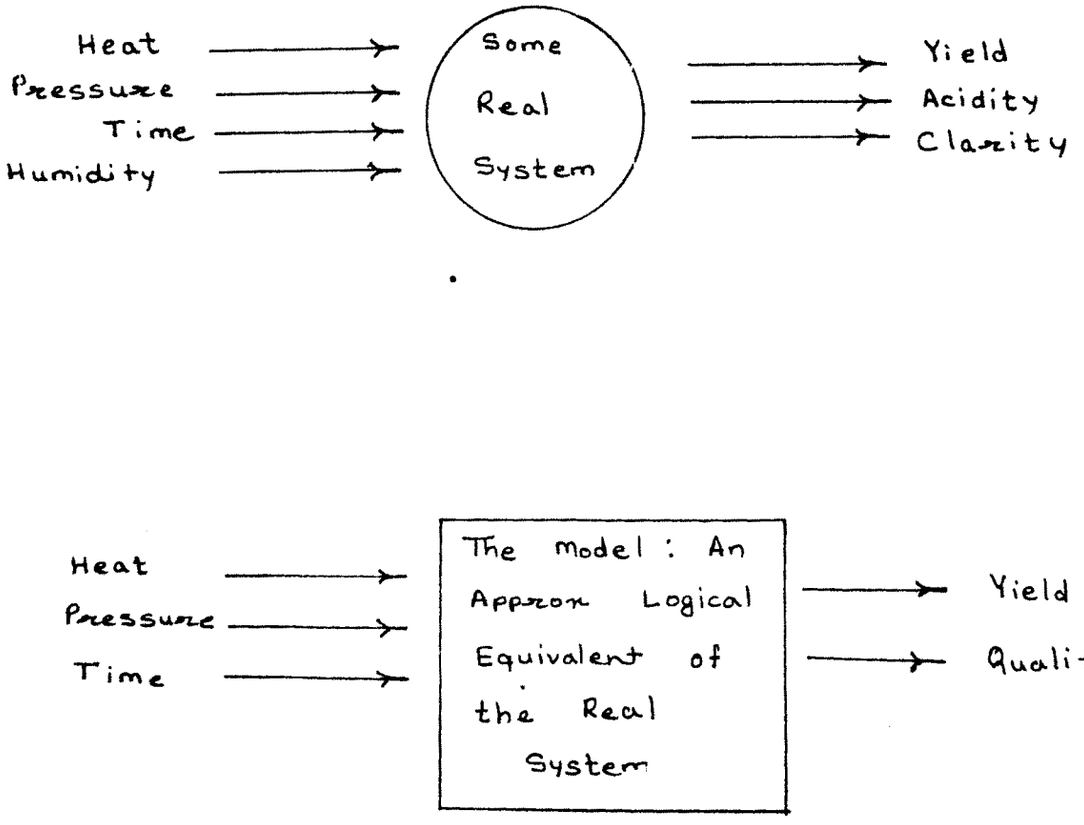


FIGURE NO.3

Typical Examples of Real System Simulation

It appears easy to work on each step of above sequence but in practice it requires a great deal of experience and skill.

4.4 COMPUTER AIDED LEARNING: STATES AND EVENTS

Physics and Electronics are the disciplines often require laborious mathematical manipulations and it is obvious that the computer can play a valuable role not only in research but in teaching as well. Complicated mathematical equations can be solved rapidally. It provide students a powerful tool to investigate the physical principles lying behind the mathematics without becoming confused by mathematical details.

The oldest reference of the working in CAL is found in 1971. Prof.Lansky et al wrote a series of experiments inorder to investigate the learning of small groups in comparison with individual learners using the method of programmed instructions in both the cases (quoted in reference 4).

A number of simulation and modelling packages have been developed by the physics department, University of Surrey. A program called "TRANS" calculates the transmission coefficients for electrons incident upon a potential barrier. This program helps in visualising the effects of varying the electron energy and the barrier width and gives the output data in tabular form. "MCOIL" is another program which simulates a moving coil meter and calculates the displacement when current passes. 'STRAIN' is a package that investigates the behaviour of elastic deformation in two dimensions defined by four tensor components.

'COUNTERS' is an experiment simulation package, simulating the radiation experiment. The objective behind designing 'COUNTER' package was that the student should gain skill in experimental strategy and familiarity with the significance of various counting techniques. All these packages are interactive in nature, model is implicit in them. While handling the package, student is aware of the model but usually has no opportunity or facility for modifying it. CAL is also used as an effective tool to understand Natural language also. Natural language is the language, used for printing, displying etc. rather than speaking. Ahmad K.[1] et al from University of Surrey have worked extensively on the application of CAL in natural language understanding. The range of natural languages covered is from German, French, Spanish, Russian upto old church Slavice, Arabic and Chinese. Initial work was based on purely visual displays with provision for input/output in two scripts. But recently an American language course has been developed in hich not only visual bilingual input/output (in American and in English) is available, but the computer can also provide audio output with good sound quality. The authors concluded that the drill exercise is fun if it is based on a computer. The students enjoyed the computer based packages and usually claim to learn as much as the package intends.

Computer simulation is a very essential tool in modern biology teaching. The modern biology is characterized by an

increasing mathematization trend. So computer simulation is the only best solution to partially replace the difficult experimental methods. Wedekind J.P.E.[13] worked on three applications of CAL in Biology. The examples are enzyme kinetics, modelling of population and modelling of biological rythms etc.

Movies have also their effect in learning science. Kaiser[9] has developed a program for a 3D movie that illustrates Kinetic Gas theory. Programs have also written for animation of Mickey Mouse[14] with $Y_O - Y_O$. So movie can be a valuable tool in entertainment as well as in learning physics. The important significance of computer generated movies in teaching physics is that they teach physics in an exact and at the same time in an unmathematical way. So it is argued that computer generated movies are valuable in teaching physics to people with poor mathematical background. Such movies are developed at the University of Karlsruhes Germany.[6] Other problems on which movies have been developed are the vibrating string problem, modelling of evaporation of fluid, scattering of the quantum mechanical probability $p(x)$ on the potential step.

Computer Graphics is particularly useful in understanding Quantum mechanics, because it uses abstract concepts (like wavefunction, effective potential, scattering phase etc.) which are not measurable or atleast inaccessible by students experiments but can be easily calculated and displayed on computer. Brandt S.[2] and et al, have developed a program 'SQUASH' (Stationary Quantum Mechanics in Schrodinger and

Heisenberg formulation). You can feed numeric input and obtain numeric as well as graphic output. The program is specifically written for time sharing system. The complex problems like with spin, problems with cylindrical or elliptic symmetries or quantum mechanics in Heisenberg formulation have also been tackled in this work.

Interactive dialog capability, portability, memory and speed efficiency are all desired qualities of a computer graphics package. These all qualities are achieved in the simulation programs "PARTICLES and FIELDS" developed by Hermann Schneider[11]. Extensive use of graphics input (GIN) is made in these programs. These programs have the capability to solve initial value problems by consecutive reduction of intervals on the display screen without going into the numbers. The program offers various aids for visualizing three dimensional objects including stereo pair of views for 3-D object.

Some experiments in physics are very difficult to realize in the laboratory. One such experiment is double slit experiment, using low intensity of light. This is the best situation to take help of computer simulation. The principle lying behind the working of simulation program is the probability interpretation of the 'interference pattern'[5] on the screen by single photons in the experimental device. By pressing the button on the key-board one can simulate the light

source, the terminal display simulates the screen with multipliers in the experiment. Two programs have been developed in FEOLL, one of them for university level, enables the student to design the experiment himself by choosing the parameters deliberately, whereas the other program, for secondary level, has fixed parameters and is divided into seven sections varying the experiment.

Yet another interesting development in CAL is it's application into 'vibrations and waves'[3] course being taught to first year students of varied scientific background. Computer with its enormous ability of calculation with speed and precision makes the lengthy and tedious calculations easy. Packages have been developed in for teaching the energy of Simple Harmonic Motion, the combination of oscillators, phasor diagram,s forced oscillations etc.

4.5 PROBLEMS WORKED OUT IN THIS CHAPTER

In this chapter two packages are developed. In the first package, the problem is taken from the ballistics and the second problem is taken from process control.

First package, fully illustrates the concept of projectile. Second package fully simulates proportional control. You might have really experienced the difficulty in explaining both the concepts in classroom. One has to go through rigorous mathematical calculations to explain the concepts. It is bit

difficult, to visualise the concept for a student with poor mathematical background. For the problem of projectile, it is almost impossible to arrange a laboratory practical. One can arrange the practical of proportional control, but it is very difficult to observe the effect of variation of setpoint and proportional band. So CAL is the best solution for both the cases.

4.6 PROJECTILE SIMULATION

4.6.1 THEORETICAL PERSPECTIVES

A particle, moving under the combined effect of vertical and horizontal forces is called a projectile. The motion of a projectile is a concept from the science of 'ballistics'. The motion of the projectile in the barrel of the gun belongs to interior ballistics and the motion from the endpoint of the gun upto the falling point belongs to exterior ballistics. Often we are interested in exterior ballistics. So exterior ballistics is simulated here. The resistance of air is neglected because it is complicated in nature.

When a particle is projected upwards at some angle (but not vertical) we find that the particle traces some path in the air and falls on the ground at a point other than the point of projection. Study of motion of the particle, reveals that the velocity of the particle can be decomposed into two components namely vertical and horizontal. Vertical component projects the

body vertically upwards and the horizontal component decides the range. Combination of both vertical and horizontal components give the particle, a parabolic path to move on it. Horizontal component remains constant while the vertical component of the motion is always subjected to gravitational acceleration.

4.6.2 TERMINOLOGY

1 Projectile : Particle moving under the combined effect of vertical and horizontal forces.

2 Trajectory : Path traced by the projectile in the space.

3 Velocity of projection : Velocity with which a particle is projected.

4 Angle of projection : Angle with the horizontal, at which the proejctile is projected.

5 Time of flight : Total time taken by a projectile, to reach maximum height and to return back to ground.

6 Range : Distance between the point of projection and the point where the projectile strikes the ground.

4.6.3 MOTION OF A BODY THROWN HORIZONTALLY INTO THE AIR

A body is placed at A and is thrown horizontally into the air with a velocity (V). Please refer to figure number 4 on page No.

Velocity of the body can be decomposed into two velocities viz. horizontal velocity and vertical velocity due to gravitational aceleration. The resultant velocity of the body

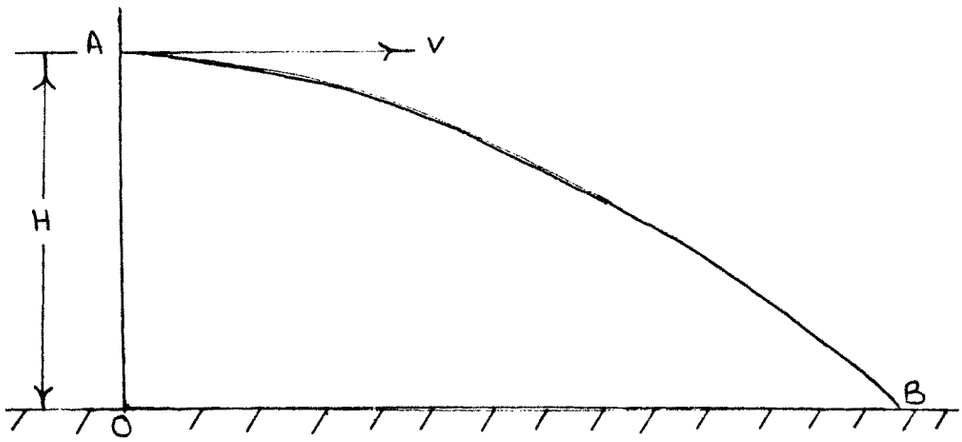


Fig.4 : Body Thrown Horizontally

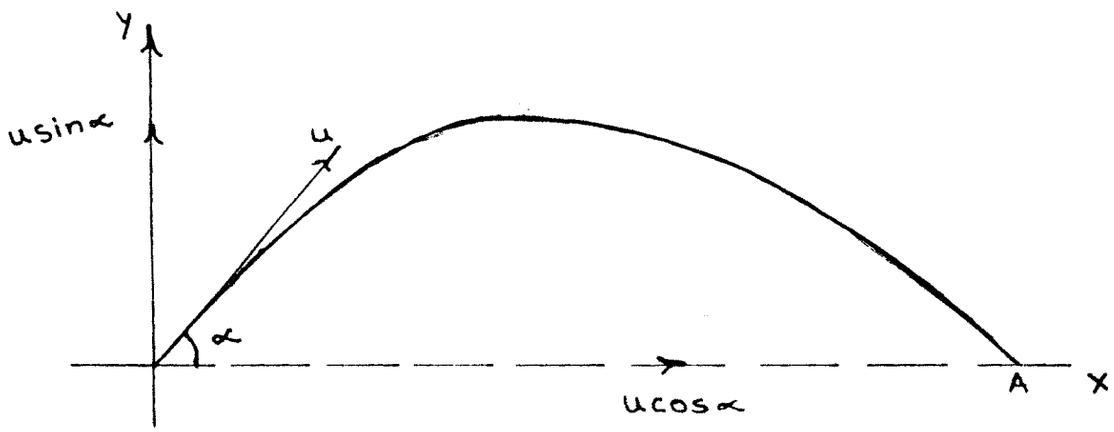


Fig.5 : Projectile on a Horizontal Plane

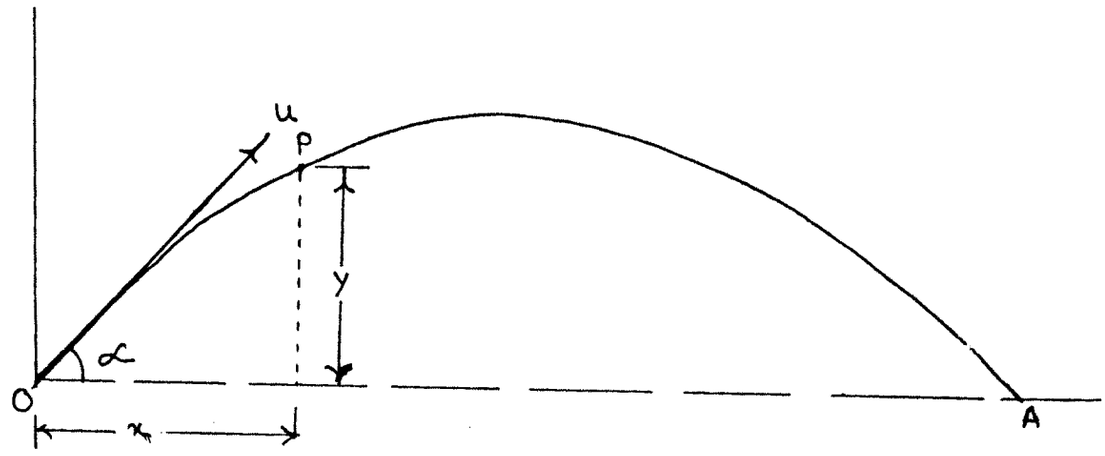


Fig.6 : Path of a Projectile

will be the vector sum of the two components. The horizontal component of the body remains constant whereas the vertical component is always subjected to gravitational acceleration. Thus the time taken by the body to reach the ground is calculated from the vertical component of the velocity, whereas the horizontal range is calculated from the horizontal component of the velocity. The velocity with which the body strikes the ground (at position B in figure), will be the resultant of horizontal and vertical velocity.

4.6.4 MOTION OF A PROJECTILE

Let alpha be the angle of projection.

u be the horizontal velocity in m/sec.

The projectile is projected from point O as shown in figure number . Now resolving the velocity into its vertical and horizontal components.

$V = u \sin \alpha$

$H = u \cos \alpha$

vertical component ($U \sin \alpha$) is subjected to retardation due to gravity. The particle will reach maximum height, when the vertical component becomes zero. Then the particle will come down, owing to the acceleration due to gravity.

The horizontal component ($u \cos \alpha$) will remain constant since there is no acceleration or retardation except the air resistance. The path of the prjectile in air will be decided by the combined effect of horizontal and vertical components.

It falls at some point A, other than the point of projection 0.

4.6.5 EQUATION FOR THE PATH OF A PROJECTILE

Please refer to figure number 5.6. In this figure 0 is the point of projection. α is the angle of projection u is the velocity of projection. The particle will move along certain path OPA, in the air, and will fall down at A. p is the position of particle, after t seconds. x and y coordinates for position 'p' are as shown in figure.

Now

Horizontal component = $u \cos \alpha$

Vertical compomnent = $u \sin \alpha$

$$y = u \sin\alpha\, t - 1/2\, g\, t^2$$

and $$x = u(\cos \alpha)t$$

or $$t = \frac{\alpha}{u \cos}$$

Substituting the value of t in equation

$$y = u \sin \left(\frac{x}{u \cos \alpha}\right) - 1/2\, g \left(\frac{x}{u \cos \alpha}\right)^2$$

$$y = x \tan \alpha - \frac{g x^2}{2 u^2 \cos^2 \alpha}$$

This is clearly the equation of a parabola, therefore the equation of trajectory is also a parabola This equation is implemented in the present work.

Other equations for checking the validity of simulation are as follows :

1. The equation for time of flight of a projectile on a horizontal plane is given by

$$t = \frac{2\,u\,\mathrm{Sin}\,\alpha}{g}$$

2. The equation for the horizontal range is given by

$$R = \frac{u^2\,\mathrm{Sin}\,2\,\alpha}{g}$$

The range will be maximum when

$\mathrm{Sin}\ 2\ \alpha = 1$

$2\ \alpha = 90^{\circ}$ or $\alpha = 45$

$$R = \frac{u^2\,\mathrm{Sin}\,90}{g} = \frac{u^2}{g}$$

3. Maximum height of a projectile on a horizontal plane is given by,

$$H = \frac{u^2\,\mathrm{Sin}^2\,\alpha}{2g}$$

4.6.6 SOFTWARE IMPLEMENTATION

The program starts with inclusion of the necessary files using include directive. The initial position of the gun is defined by X GUN and Y GUN. Next module of the program is "Functions" in which functions are defined. The purpose of various functions is given below.

gun : Draws the cannon and gun

tra : Computes the trajectory

blast : Creates illusion of blast

asp-rotate-about : Called in polarline function for aspect ratio considerations.

rotate : rotates ther gun.

changeangle : Changes the angle of gun within some maximum and minimum limits.

initialset : Displays menu.

fly : starts the firing.

In the main module, the initialization of the graphics system is done. With the 'circle' statement, a ball is created and is filled with 'setfillstyle'. The cannon ,gun and ball are placed at their location.

The choice module waits for the proper key and executes the command associated with that key. If only 'ENTER' is pressed then it shows the projectile trajectory for that angle. If you have to change the angle then press key 'a' followed by key '+' for increasing and key '-' for decreasing. After setting the angle, press enter key the ball appears. Again press enter the ball flies.

The next part of the program is function 'Fly'. It shows the ball flying over the calculated path (which is parabolic one). The other tasks of this function is leaving behind the trace of the ball by dotted lines and constantly displaying the height and distance when the ball travels. The display settle down when the ball reaches ground.

The 'gun' function draws the cannon and gun. It also varies the position of the gun according to the user entered keys.

Actual calculation of trajectory is done by the function 'tra'. The equation implemented here is

$$y = Ynit - x \tan(ang.) - \frac{g\,x^2}{2u^2 \cos^2(ang.)}$$

where the symbols have their usual meaning.

'Blast' function creates an illusion of blast. Appearing and disappearing of arcs with increasing radius accompanied by beep from the speaker creates the effect of blast.

The function asp-roate-about rotates the gun considering the aspect ratio of the screen. If the aspect ratio is not considered while rotating the gun, the straight line of the gun will appear as steps. In this function first the aspect ratio of the screen is taken. Then the origin is shifted and the coordinates are scaled according to aspect ratio. Rotation function is now called to rotate the line. Before displaying on the monitor the coordinates are again scaled down. The rotation function performs all the tasks for rotation.

The function 'changeangle' changes the angle of the gun. The angle is increased or decreased by pressing the key '+' and key'-' respectively.

Flashing of menus and other related messages is done by the function 'initialset' (Refer to the diagram of sample screen on page number 134).

Refer to listing of program on page number. 115.

4.7 SIMULATION OF PROPORTIONAL CONTROL

4.7.1 PROPORTIONAL CONTROL

Proportional control is the natural extension of multistep controller. Here the final correcting device is not forced to take an all-or-nothing position. Instead, it takes a continuous range of possible positions. The exact position that it takes is proportional to the error signal. Stated in other way, there is a linear relationship between output of a controller to its input. The word proportional is correctly applied to this mode because the amount of correction is in proportion to the amount of error.

4.7.2 PROPORTIONAL BAND

Proportional band is the percent of full controller range by which the measured value must change in order to cause the correcting device to change by 100%. A good proportional controller[8] should have an adjustable proportional band, usually variable from few percent upto a few hundred percent.

4.7.3 THE EFFECT OF PROPORTIONAL CONTROL

Proportional control eliminates the permanent oscillation that always acompanies on-off control. (This is not true for very small proportional band). But proportional control works well only in systems where the process changes are quite small and slow. This type of control helps if the disturbance occurs slowly, becuase then the proportional band can be adjusted quite

narrow, since there is not much oscillation produced by a slow process change. But narrow proportional band means more oscillations.

4.7.4 MATHEMATICAL EXPRESSION :

Proportional mode can be expressed as

$$P = K_p E_p + P_o$$

where,

K_p = Proportional constant between error and controller output (%/%).

P_o = Controller output with no error (%).

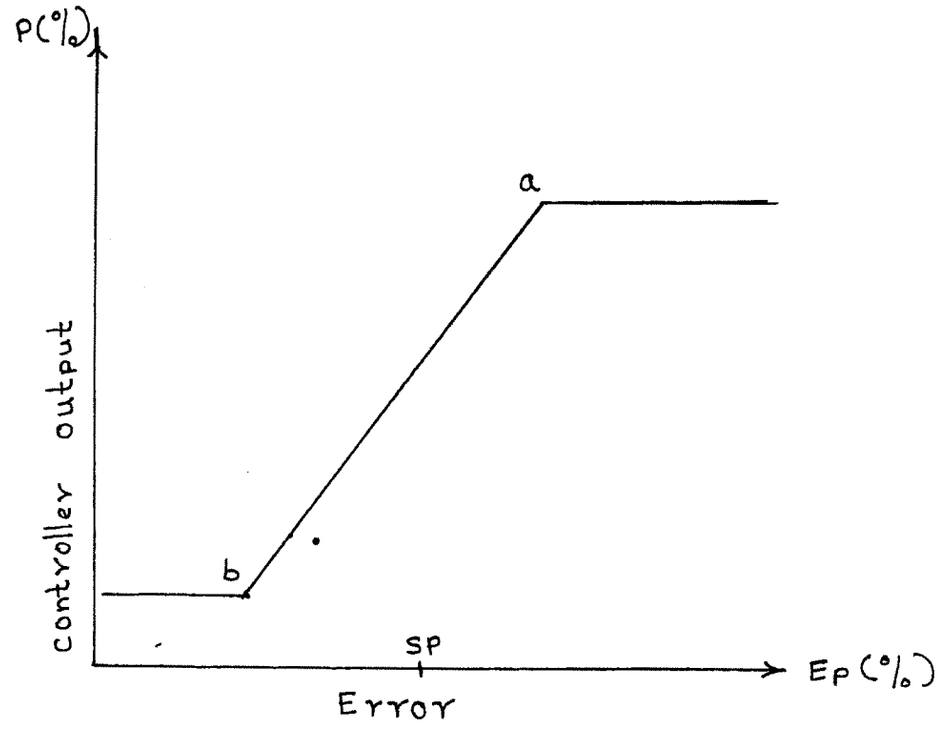
The proportional band is given by $100/K_p$. The controller output for errots exceeding the proportional band is saturated at either 100% or 0% depending on the sign of the error.

4.7.5 OFFSET IN PROPORTIONAL CONTROL

One drawback of the proportional control is that it produces a permanent residual error in the operating point of the controlled variable when a change in load occurs. This error is referred to as offset. By keeping K_p large (i.e. smaller proportional band) the offset can be minimized. In the present simulation offset is not considered.

4.7.6 LOGIC BEHIND SIMULATION

Proportional control can be graphically represented as shown in Fig.



The slope of the straight line ab depends upon the proportional band. Traditionally error (difference between measured variable and setpoint) is represented on x axis. But in this work measured variable is represented on x axis, while the location of setpoint is shown on x axis by 'SP'. Both the measured variable and controller output have the variation range from 0 to 100.

On execution of the program one can see the graph of proportional control with default values. For this default case the values of setpoint and proportional band are 50 and 40. The controller output at setpoint is taken as 50.

Now to observe the variation of controller output with measured variable, there must be some facility to change the

measured variable. This can be done by using the left and right arrow keys (direction keys). With left key, one can increase the measured variable upto 100, while with the right key one can decrease the measured variable upto 0. For better visualisation, a horizontal bar just below the x axis is place. with 0 measured variable, the bar is empty. As you go on increasing he measured variable, it goes on filling. At the sametime a numerical display shows the value of measured variable. In the same manner, a vertical bar is placed parallel to the y axis which shows the variation of controller output. Now one can clearly boserve that the controller output varies proportionaly in the range of proportional band. Outside the range of proportional band, there is no variation of controller output value.

Better understanding of the proportional control can be made by observing the effects of variation of proportional band and setpoint. By pressing key 'F2' one can change the setpoint. The user has to give the new setpoint interms of numbers. By pressing key 'F3' one can change the proportional band, which is also to be given by numerical value. Press key'F8', you will be on DOS prompt. Refer to listing of program on page number 122.

```
/*  This program is developed for simulating  a  PRO
    JECTILE It can be used as an effective tool  for
    teaching the concept of  'PROJECTILE-TRAJECTORY'
    which   is   a  basic  concept   in   MECHANICS.
    If you want to change the PROJECTION ANGLE  then
    press  key 'a', followed by key '+' to  increase
    the  angle  &  key '-' to decrease.  Press  key
    'ENTER' twice to fire. After enough  drill-work,
    press key 'Esc' to quit.For hard-copy press  key
    'p'.

    /* PROGRAM STARTED ON : 02-12-1993
       FINAL VERSION ON   : 30-12-1993        */

   /*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*/

   #include <stdio.h>
   #include <conio.h>
   #include <graphics.h>
   #include <stdlib.h>
   #include <dos.h>
   #include <math.h>
   #include "gprnt.h"

   #define YGUN 200
   #define XGUN 100

   /*$$$$$$$$$$  FUNCTION  MODULE $$$$$$$$$$*/
   void gun(int *,int *);
   int tra(int);
   void blast(int ,int);
   void polarline(int ,int ,double ,double );
   void asp_rotate_about(int ,int ,double , double ,
                                  double *, double *,double

   void rotate(double ,double ,double *,double *,double );
   void changeangle(int);
   void initialgot(void);
   void fly(void);

   double fireangle = 30;
   double u = 50;
   int yinit;
   int init;
   void *ball;

   /*::::::::: MAIN  MODULE  ::::::::::::::::::*/

   main()
   {
           int x,y;
           int drive = HERCMONO;
           int mode = DETECT;
           unsigned size;
           int op;
```

```
initgraph(&drive,&mode,"");

circle(20,20,10);                    /* Ball    */
setfillstyle(SOLID_FILL,1);
floodfill(20,20,1);

size = imagesize(10,12,30,28);
                                  /*  store the ball*/

ball = malloc(size);
if(!ball) exit(0);
getimage(10,12,30,28,ball);
putimage(10,12,ball,XOR_PUT);

y = YGUN;
x = XGUN;
gun(&x,&y);
putimage(x-10,y-10,ball,XOR_PUT);


/*  canno ball */

init = x;
yinit = y+10;


/*++++++++   CHOICE  MODULE +++++++++ */

initialset();
changeangle(0);         /* will display the angle

                        /* & quit */

for(;;)
æ
op = getch();

switch(op)
æ


case 'a':  putimage(x-10,y-10,ball,XOR_PUT);
changeangle(1);
x = getx();  y = gety();
putimage(x-10,y-10,ball,XOR_PUT);
init = x;
yinit = y+10;
break;

case 27 :  closegraph();
exit(0);

case 13 :  fly();
break;
```

```
case 'p' : ghardpict(1,1,640,320);
break;

default :  break;
ß

ß

ß

/*  start  the  firing   */

void fly(void)
æ
int x,y;
char distÆ6A;
char htÆ6A;


for(x = init; x<700 ; x=x++)
æ

y = tra(x-init);
if(y>330) æ
blast(x,y-10);
setcolor(1);
outtextxy(x-7,y,"D");
outtextxy(x,y,dist);
outtextxy(x-7,y+10,"A");
itoa(fireangle,ht,10);
outtextxy(x,y+10,ht);
break;
ß
putimage(x-10,y-20,ball,XOR_PUT);
delay(10);
putimage(x-10,y-20,ball,XOR_PUT);
if(!(x%10))
æ
putpixel(x,y-10,1);
setcolor(0);
outtextxy(600,32,dist);
outtextxy(600,20,ht);
setcolor(1);
itoa(x-XGUN,dist,10);
outtextxy(600,32,dist);
itoa(YGUN - y,ht,10);
outtextxy(600,20,ht);
ß

ß

ß
```

```
/*   THE  CANNON    */

void gun(int *p,int *q)
æ
int XG,YG;
XG = *p;
YG = *q;
polarline(XG,YG,60,fireangle);
*p = getx();
*q = gety();

line(XG-80,YG-11,XG-40,YG-11);
line(XG-80,YG-11,XG-80,YG+20+10);
line(XG-80,YG+20,XG+20,YG+20);
line(XG-80,YG+20+10,XG+20+10,YG+20+10);
line(XG+20,YG+20,XG+20+10,YG+20+10);

arc(XG-40,YG+20 ,0,90,40);
å


/*  COMPUTES   TRAJECTORY   */
int tra(int x)
æ
int y;
double temp;
double con ;
double g = 9.81;
double pi = (double) 22/7;
double ang;

ang = (double)(fireangle * pi)/180;
temp = cos(ang);
temp = 2 * u*u*temp * temp;
temp = (double) (g * x * x)/(temp);
con = tan(ang);
temp = (x*con) - temp;
y = (int) yinit - temp;
return y;
å


/*  CREATES  THE   BLAST  */
void blast(int a,int b)
æ
int r;

for(r = 10; r < 50 ; r = r + 10)
æ
setcolor(1);
arc(a,b ,0,180,r);
```

```
    sound(1000);
    delay(60);
    nosound();
    setcolor(0);
    arc(a,b ,0,180,r);
á   á


     void asp_rotate_about(int xcn,int ycn,double x,
       double y, double *nx, double *ny,double angle)
æ
            double xrt;          /* shifted points rotated*/

    double yrt;
    int xasp,yasp;
    char color;
         getaspectratio(&xasp,&yasp);
    x = x - xcn;        /* shift origin to  xcn,ycn  */
    y = y - ycn;
            x = x * xasp;     /* scale according to */
    y = y * yasp;            /* aspect ratio */

    rotate(x,y,&xrt,&yrt,angle);     /* rotate about */
                                     /*     origin */
            xrt = xrt / xasp;       /* back  to original
                                       scale */
    yrt = yrt / yasp;
            *nx = xrt + xcn;       /*  shift back the ori
                                        gin */
                           /*    to previous place */
    *ny = yrt + ycn;        /*push the new co-ordinates*/
            á
  /*MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM*/


            /* POLAR  LINE  */
            void polarline(int x,int y,double mag,double
                                              ang);
    æ
    double xn,yn;
    ang = (double) (22 * ang * -1)/(180 * 7);
                          /* ø__ converts to radian */
    asp_rotate_about(x,y,x+mag,y,&xn,&yn,ang);
    line(x,y,(int)xn,(int)yn);
    moveto((int) xn,(int)yn);
    á
           /*++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* ROTATE  THE POINTS X,Y
pointers to x,y are sent. Addresses where  new
coordinates are placed is sent.All the
coordinates are double.  Angle is in radians.*/

void rotate(double x,double y,double *nx,
            double *ny,double angle)
æ
*ny = x * sin(angle)  +  y * cos(angle);
*nx = x * cos(angle)  -  y * sin(angle);
ä

/*MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM*


/* _____ CHANGE ANGLE _____________________ */

void changeangle(int staflg)
æ
char op;
int fr;
char angnoŒ5A;

fr = fireangle;
itoa(fr,angno,10);
outtextxy(XGUN-78,YGUN+49,angno);

if(staflg == 0) return;     /* if the prog is run
                               for first time */

rectangle(XGUN-82,YGUN+39,XGUN-28,YGUN+71);

for(;;)
æ
op = getch();

setcolor(0);                    /*  remove the line

polarline(XGUN,YGUN,60,fr);
outtextxy(XGUN-78,YGUN+49,angno);

switch(op)                          /*  action   */
æ
case '+' : fr++;
if(fr>70) fr--;
break;

case '-' : fr--;
if(fr<10) fr++;
break;

case 13  : fireangle = fr;
setcolor(0);


rectangle(XGUN-82,YGUN+39,XGUN-28,YGUN+71);
```

```
setcolor(0);


rectangle(XGUN-82,YGUN+39,XGUN-28,YGUN+71);
setcolor(1);

polarline(XGUN,YGUN,60,fr);
outtextxy(XGUN-78,YGUN+49,angno);
return;

å

setcolor(1);                          /*  line  */
polarline(XGUN,YGUN,60,fr);
itoa(fr,angno,10);
outtextxy(XGUN-78,YGUN+49,angno);

å           /* for  */

å



void initialset(void)
æ

outtextxy(200,2,"PROJECTILE TRAJECTORY");
rectangle(XGUN-80,YGUN+40,XGUN-30,YGUN+70);
outtextxy(XGUN-80,YGUN+75,"Angle");
outtextxy(XGUN-52,YGUN+45,"+");
outtextxy(XGUN-52,YGUN+60,"-");


outtextxy(520,32,"DISTANCE :");
outtextxy(520,20,"HEIGHT   :");

outtextxy(XGUN-80,YGUN+120,"A-Change angle
       Enter - fire      Esc - Abort   p - Print);


å
```

```
/* PROPORTIONAL CONTROL SIMULATION */
/* EFFECTIVE IN TEACHING GRAPHICAL RELATION */
/* IN CONNTROLLER O/O & ERROR. EFFECT OF PROPOR
   TIONAL BAND & SETPOINT VARIATION IS ALSO
   ILLUSTRATED */
   DATE :- 01-01-1994
```

```
#define X1   150          /*     X1___X2     X3                      X4 */
#define X2   180          /*  Y1 ø    ø       ø                        */
#define X3   220          /*     ø    ø       ø                        */
#define X4   420          /*     ø    ø       ø                        */
                          /*     ø    ø       ø                        */
#define Y1   80           /*  Y2 ø___ø       ø_____________           */
#define Y2   180          /*                                           */
#define Y3   200          /*  Y3 -   -      ø----------------ø*/
#define Y4   230          /*                 ø                 ø*/
#define XTI  100          /*  Y4 -   -      ø----------------ø*/

#define YTI 10          /*   down window position       */
#define HOT 8
#define OUTLINE 1       /*  color of axes                    */
#define CURVE   1       /*  Color of curve                   */
#define HVLINECOL   1
#define BARCOL   1

#include <graphics.h>
#include <stdio.h >
#include <conio.h >
#include <stdlib.h>
#include <math.h>
#include <dos.h>

#include "defn.h"
#include "import.h"


/*  Functions  */
void connecting_line(int,int);
void draw_curve(int ,int );
void put_vbar(int,int);
void put_hbar(int,int);
void create_vhbar(void);
void change_sp(float *, float  *,int ,int);
void change_pb(float *, float  *,int);
void remove_block(int,int);
/* void svach(char *);       */
/* char *ask_user(char *);  */

  static void *vbar;          /* for filling the bars  */
  static void *hbar;
```

```
int sp;    /*  Set Point                */
int pb;   /*  Praportional Band       */

int cclr = 1;
main()
  {
      int drive;
      int mode;

      int x,y;          /*  screen  co_ordinates  */
      float pout;       /*  P  Controllers Output*/
      int min;      /*  M  measured Value      */

      int poutsp;     /*  Output at S.P          */
      float a,b;    /*  Knee of the curve      */
      char *nm1,*nm2;  /*  display parameters */
      int  yold;
      int i;

       union•inkey æ    /*  to split the asci & scan
                        codes      */
       char  chÆ2A;
             /*  returned by the bioskey() */
       int i;
       å c;


      / ***************************************/

       nm1 = (char *)calloc(6,2);
       nm2 = (char *)calloc(6,2);

     /* INITIALISATION  OF  GRAPHICS  MODE  */
       drive = DETECT;
       mode = 1;
       initgraph(&drive,&mode,"");

        create_vhbar();


        /* SET TEXT  ATTRIBUTES     */
        setcolor(HOT);

        settextstyle(DEFAULT_FONT,VERT_DIR,1);
       outtextxy(X1-35,Y1-25,"controller output");

       settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
 outtextxy(XTI+70,YTI,"PROPORTIONAL CONTROLLER");

outtextxy(X3+120,Y4+22,"measured parameter ->");


outtextxy(10,332,"F2  -  Change the  setpoint  F3-
Change Praportional Band   F8 -  QUIT ");
```

```
/* MAIN  LOOP    ****************************/

/*   INITIALISATION   of the loop  */

  pout = 0;   min = 0;
  x = X3;  y = Y2;

  sp     = 50;
  poutsp = 50;
  pb     = 40;

  outtextxy(300,40,"Praportional Band :");
  outtextxy(500,40,"40");

  /*  AXIS  and     BARS  */
  setcolor(OUTLINE);
  line(X3, Y1,X3,Y2);
  line(X3,Y2,X4,Y2);
  rectangle(X1,Y1,X2,Y2+2);
   rectangle(X3-3,Y3,X4,Y4);

   line(sp*2+X3,Y2,sp*2+X3,Y2+5);
   outtextxy(sp*2+X3+4,Y2+4,"SP");


   a = sp - (pb*poutsp/100);
   b = sp + (pb*poutsp/100);

  draw_curve((int)a,(int)b);
      /*  Default curve       */

  connecting_line(x,y);  /*Put connecting line */

  put_vbar(x,Y3);
  put_hbar(X1,y);

 /* loop starts here          */

  for(;;)æ

  while(!bioskey(1));
      /*  waits till a key is pressed */
  c.i  = bioskey(0);
      /*  gets the asci & scan code   */


  connecting_line(x,y);
    /*  Remove the connecting lines */

   setcolor(0);
        /* erase previous nos  */
   outtextxy(x, Y4+8,nm1);
   outtextxy(X1-30,y,nm2);
   setcolor(1);
```

```
if(!c.chÆ0Å)  æ

 switch(c.chÆ1Å)  æ


/*  ________CURSOR MOVEMENT_____________*/
    case 75: min = min-1;
               /* left  */

   if(min<0) æ
              min = min++;
              break;
              å
   put_vbar(x,Y3);
       /* The previous bar removed */
   break;
          •
   case 77: min = min+1;
   if(2*min+X3>X4)
                æ
                min = min--;
                break;
                å
   put_vbar(X3 + 2*min,Y3);
         /* modified & then put */
   break;

   case 66: closegraph();
   exit(0);

   case 60: remove_block(sp*2+X3-1,Y2+1);
   remove_block(sp*2+X3+4,Y2+4);
   remove_block(sp*2+X3+8,Y2+4);
   change_sp(&a,&b,pb,poutsp);
   break;


   case 61:
   remove_block(500,40);
   remove_block(510,40);
   change_pb(&a,&b,poutsp);
   break;


      å
       å

  if(min<a) pout = 0;
   else
    if(min>b) pout = 100;
     else
       pout = (min - a)*100/pb;
```

```
        yold = y;

        x = X3 + 2*min;
        y = Y2 - pout;

        if(yold>y)
           {
        for(i = y; i < yold ; i++)
        put_hbar(X1,i);
           }
        else    for(i = yold; i < y ; i++)
        put_hbar(X1,i);

      /*  DISPLAY  CURSOR  POSITION    */


        setcolor(1);
          /*  now display new coordinates */
        itoa((int)min,nm1,10);
        outtextxy(x,Y4+8,nm1);

        itoa((int)pout,nm2,10);
        outtextxy(X1 - 30,y,nm2);

         setcolor(HVLINECOL);

         connecting_line(x,y);
          /*  Put the connecting lines */

             {
              }


 /* *     F U N C T I O N S      *   * */

/* line from measured value to the curve, to
    the controller output  INPUT -  point on
    the CURVE
      void connecting_line(int x,int y)
       {

        setcolor(cclr);
        cclr  = !cclr;

       /*   vertical line   */
       if( ! ( (x>sp*2+X3-1)&(x<sp*2+X3+23) ) )
           line(x,Y3-1,x,Y2+1);
       if(y!=Y2)    line(x,Y2-1,x,y+1);

          /*   horizontal line   */
       if(y != Y1)line(x-3,y-1,X3+1,y-1);
        else     line(x-3,y,X3+1,y);
       line(X3-1,y-1,X2+1,y-1);   }
```

```
/* CURVE for the P controller,  Points to be
   INPUT :   Knees of the  Curve    */
   void draw_curve(int a,int b)
    æ
      a = X3 + 2*a;
      b = X3 + 2*b;
      line(X3,Y2-1,a,Y2-1);
      line(a,Y2-1,b,Y1-1);
      line(b,Y1-1,X4,Y1-1);

      setcolor(HVLINECOL);
        å


     /*  Bars for Filling up the bars  */
     void create_vhbar()
       æ
      vbar = malloc((X2-X1)*4);
      hbar = malloc((Y4-Y3)*4);

       setcolor(BARCOL);
       line(10,Y3+1,10,Y4-1);
       line(11,Y3+1,11,Y4-1);
       getimage(9,Y3,12,Y4,vbar);
        putimage(9,Y3,vbar,XOR_PUT);

      line(X1+1,10,X2-1,10);
      getimage(X1,9,X2,11,hbar);
      putimage(X1,9,hbar,XOR_PUT);

       å


      void put_vbar(int x,int y)
        æ
      putimage(x-4,y,vbar,XOR_PUT);
        å

      void put_hbar(int x,int y)
         æ
       putimage(x,y,hbar,XOR_PUT);
         å


/*************************************/
/* CHANGE SET POINT*/

 void change_sp(float *ap, float *bp ,
         int pb, int poutsp)

  æ
     float a,b;
     char *newsp;
```

```
        newsp = (char *)malloc(5);

        a = *ap;
        b = *bp;

       setcolor(0);
       draw_curve((int)a,(int)b);
       setcolor(1);


        newsp = ask_user("NEW SETPOINT");
        sp = atoi(newsp);

        setcolor(BLUE);
        line(sp*2+X3,Y2,sp*2+X3,Y2+5);
        outtextxy(sp*2+X3+4,Y2+4,"SP");


        a = sp - (pb*poutsp/100);
        b = sp + (pb*poutsp/100);

       draw_curve((int)a,(int)b);
              /*  new curve        */
        *ap = a;
        *bp = b;
        free(newsp);

         a


/*****************************************/
 /*    CHANGE    PRAPORTIONAL BAND     ******/

void change_pb(float *ap, float *bp , int poutsp)
       æ
        float a,b;
        char *newpb;

       newpb = (char *)malloc(5);

       a = *ap;
        b = *bp;

       setcolor(0);

       draw_curve((int)a,(int)b);
       setcolor(1);


      newpb = ask_user("NEW P_BAND");
      pb = atoi(newpb);

      setcolor(BLUE);
     outtextxy(300,40,"Praportional Band :");
```

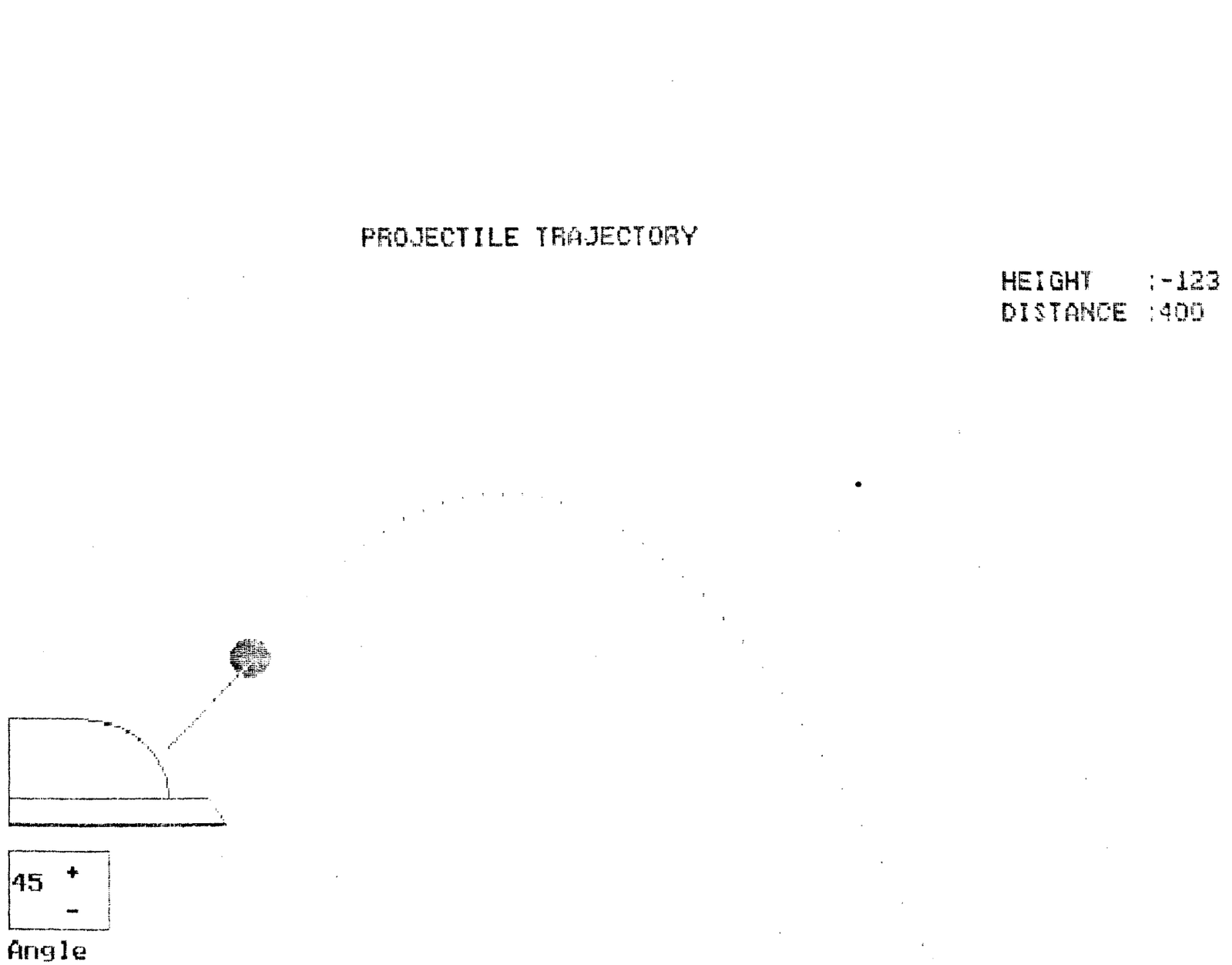
```
outtextxy(500,40,newpb);

  a = sp - (pb*poutsp/100);
  b = sp + (pb*poutsp/100);

  draw_curve((int)a,(int)b);
        /*  new curve        */
  *ap = a;
  *bp = b;


  free(newpb);

   å

         •

   void remove_block(int x,int y)
    æ
   int i,j;
   for(i = x; i < 12+x; ++i)
        for(j = y; j < 8+y; ++j)
          putpixel(i,j,0);

     å
```

```
/* LISTING OF IMPORT.H    */
/*    THIS FILE IS TO BE CONNECTED WITH  PCHERC.C
/*    THE PURPOSE OF THIS HEADER FILE IS TO GET
      IN THE DATA ENTERED BY THE USER.
      THIS FILE PROMPTS THE USER IN AN  ATTRACTIVE
      WAY e. g. BY PUTTING A SOLID BAR etc.  */
      CGA WILL ENHANCE ITs   ATTRACTION
/*""""""""""" PROGRAM STARTS  HERE """""""""""" */

#include <string.h>
#include <process.h>


#define PI (22/7)
#define XB 8
#define YB 6


char *ask_user(char *);
                  /* screen prompt          */
void svach(char *);
                 /* scren read             */
void solid_box(int,int,int,int,int);
                /* block                  */
void pt_mark(int,int);
               /* point marker         */
void xor_place(void);  /* sets XOR mode*/

unsigned  char getdownbyte(int x,int y);
       /* prepare byte for download */

void  putdnbyte(unsigned char  db);
      /* display a byte  */

void  line_paint(int  ,int ,int ,int  );
      /*  solid block              */


/* __PROMPTS THE USER_____________________________*/
  /*   stores a fixed region from the screen. Puts a
       white block there. Gives Message to the user.
       */

    char *ask_user(char *prmt)
      {
        char *tmpit;
        static char *auser;
        int color;

        color = getcolor();

        auser  = (char *)calloc(10,1);
```

```
/* store the region where name is prompted */

        tmpit = calloc(4000,1);
        if(!tmpit)  exit(0);
                /*  if no space  */

        getimage(40,40,200,70,tmpit);

                 /* Prompt the message */
        solid_box(40,40,200,70,HOTBK);
        setcolor(HOT);
        moveto(50,50);
        outtext(prmt);

        moverel(5,0);  /* leave some space */
        svach(auser);   /* get in the name  */

 /*     restore the image removed for prompt  */
        putimage(40,40,tmpit,COPY_PUT);
          /* restore the image */
        free(tmpit);
        setcolor(color);
           /* restore the color */
        return auser;
        á
/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/*___GETS  THE  CHAR  STRING FROM  SCREEN _____*/
 /* echos with HOT at 130,50  */
  void svach(char *ch)
  æ
    char numb;
    char *cht;
    char i = 8;
    cht = ch;

   do æ

     numb = getch();
         if(numb == 8)æ
         *cht = NULL;
            /* back space pressed  */
         cht--;
         i++;    /* retrive , remove the old */
         moverel(-8,0); /* char */
         setcolor(HOTBK);
         outtext("Æ");
         moverel(-8,0);
         setcolor(HOT);
         continue;
         á

         if(!numb) continue; /* special key neglect
                                cont (go to end)*/
```

```
        if(numb==27)  /* if ESC key is pressed*/
        æ
          ch = NULL;
          return;
        å

         if(!i) æ   /*i reached 0;cont no break*/
      if(numb!=13)  continue;  /* entered  char  is
                                   ENTER   */
        å


      /*      MAIN     */
         *cht = numb;
         setcolor(HOT);
         if(numb  !=  13)  outtext(cht);
             /* dont display if enter is pressed */
         cht++;
         i--;
         å while(numb != 13);
         cht--;
         *cht = NULL;

          å
      /-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/


/*_____SOLID BLOCK_____________________________________*/*
    CREATES  RECTANGULAR BLOCK IN SPECIFIED  COLOR!
      WITH   BLUE BORDER.  (EFFECT CANNOT  SEEN  ON
          HGA */

     INPUTS: X1,Y1,X2,Y2;
     COLOR;

     void solid_box(x1,y1,x2,y2,c)
     int x1,y1,x2,y2,c;
       æ
            setcolor(c);
            rectangle(x1,y1,x2,y2);
            setfillstyle(SOLID_FILL,c);
            floodfill(x1+1,y1+1,c);
        å
  /*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/
```

```
        /* _____ SMALL BLOCK ____*/

    void pt_mark(int x,int y)
     æ
          xor_place();
          line(x - 2,y-3,x+2,y-3);
          xor_place();
          line(x+2,y-3,x+2,y+3);
          xor_place();
          line(x+2,y+3,x-2,y+3);
          xor_place();
          line(x-2,y+3,x-2,y-3);
          moveto(x,y);
     å
   /*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

                    •

/*_____XOR PLACE_____________________________________*/

         void xor_place(void)
          æ
           int oc;
           oc = getcolor();
           oc = !oc;
           setcolor(oc);

           å
  /*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

       /*________ LINE BOX ______*/

     void line_paint(int x1,int y1,int x2,int y2)
        æ
            int xmax,ymax,xmin,ymin,tx,ty;
            int color;
            color = getcolor();
            setcolor(HOTBK);

            xmax  -  max(x1,x2);  /* get  top-left  &
                        bottom_right corners */
            xmin = min(x1,x2);
            ymax = max(y1,y2);
            ymin = min(y1,y2);

            for(ty = ymax ; ty > ymin ; ty--)
            line(xmin,ty,xmax,ty);
            setcolor(color);

            å


       /*-*-PROGRAM ENDS  HERE*-*-*-*-*-*-*-*-*-*-*-*/
```



Sample Screen of the program for simulating projectile Gun and canon is shown

REFERENCES

1 K.Ahmad and M.Rogers, "Development of Teaching Packages for undergraduate students of German" pp.253-263

2 S.Brandt and P.Janzen, "An interactive computer program for the Univeristy teaching of Quantum Mechanics" pp.153-174

3 Margaret Cox and David Lewis, "Developing CAL for a vibrating and wave course" pp.107-116.

4 Wildenberg Detlef, "Considerations on Computer Graphics for the use by small groups" "Computer Simulation in University Teaçhing".D.Wildenberg, edition, North Holland Publishing Company,1981 pp.101-106.

5 Jit ka Fenclova Brockmeyer, "The simulation of the double slit expt.at low intensity of light" pp.117-128.

6 H.Genz, "Visulalizing Physical processes by Computer generated movies "pp.175-186.

7 Harris D."Computer Graphics and Applications," Chapman and Hall Computing 1984,pp.146-1149.

8 Johnson C.D,"Process Control Instrumentation Technology," Second Editorm John Wiley and Sons, 1982,pp.297-300.

9 F.Kaiser,"A computer Generated movie on 3-dimensional Kinetic Gas Theory" pp.187-194.

10 Khurmi R.S."A text book ofEngineering Mechanics," S.Chand and Company Ltd.,Eighteenth Edition 1989,pp.463-489.

11 Hermann Schneider "The dialog of the Physics Simulation program "PARTICLES AND FIELDS" ,pp.129-152.

12 John L. Synge and Byron A.Griffith,"Principles of Mechanics" McGraw Hill Book Company,Inc.1959 pp.138-146

13 J.P.E. Wedekind, "The Instructional use of Computer Simulation in the Teaching of Bilogy, Three examples" pp.223-235., Proceedings of the FEoLL workshop Paderborn, Germany, 28-30, Jan 1980.

14 Batty M., "Microcomputer Graphics" Chapman and Hall Computing 87 pp.300-311.

15 Sawusch M.R. and Summers T.A., "1000 things to do with your IBM PC" BPB publication JAB Books Inc.USA, First editor 1985.

N.B. : Reference numbers, 1,2,3,4,5,6,9,11 are available in the form of a book which is edited by D.Wildenberg and Published by North Holland Pubvlishing Company in 1981. The page numbers in the references are the page numbers of this book.