***Project Members:***

***Ram Agarwal - 16102A0053***

***Navanit Srisangkar- 16102A0065***

***Aaditya Rane - 16102A0066***

Data Structures Mini Project

Bubble Shooter

1.***Introduction***:

In this project, we are going to implement a very popular and widely used game “Bubble Shooter”

using concepts of data structures such as linked list and queues along with concepts of computer graphics to develop this wonderful game.

Bubble Shooter will have you addicted from the very first bubbles you shoot down. This is a timeless game that still remains challenging. Though seemingly simple enough for children, this engaging game has remained at heart for players of all ages. The goal is to remove all the bubbles from the screen by carefully aiming your own bubble to hit and knock down a group of three or more same coloured bubbles. Adjust your winning strategies in each game! The more bubbles you remove in one shot, the more points you score, and the closer you get to becoming the master bubble shooter!

2.***Problem* *Decomposition***:

In this implementation, we are using concepts of graphics to design bubbles arranged in a sequential manner with the help of linked lists and queues. The bubbles are coloured randomly and beautifully using concept of recursion for better playing. We initialize mouse and use various mouse functions to perform mouse options such as aiming the bubble and shooting them with mouse along with displaying position of mouse using coordinates. As the user shoots bubbles, the score will simultaneously change as per the number of bubbles shot.

The problem can be decomposed as follows:

1. Arrangement of bubble network on output screen and using appropriate data structures for the following along with suitable colours for each bubble positioned randomly.
2. Creating scorecard along with current score for player to observe his progress.
3. Initializing and using mouse for aiming the bubble to shoot at the network of bubbles.
4. Knocking down the bubbles which were shot by the bubble of same colour and changing score accordingly using functions defined in the program
5. After completing the game displaying the final score and other details to the user.

3.***Identifying Data Structures***

The data structures we used for implementing “Bubble Shooter” are as follows:

1. Linked List: Linked list is one of the best suitable data structure which reduce the errors caused by use of simple array. Down below is detailed description of advantages of use of Linked List in this mini project: -

* Linked list allows to connect every node(BUBBLE) to its surrounding 6 nodes and very mush reliable to make networks of bubble as small interconnected graphs with each node holding individual identity.
* Although it is not better than simple array to access any node using linked list. But on other hand it is deleting a node or making it colourless becomes easy. Which is the key point of use of linked list.
* Also since all nodes are interconnected hence operation to drop all adjacent nodes of same colour using recursion becomes easy.
* Again, if you observe in the implementation that every row contains bubbles in alternate manner. So again, use of linked list allows to connect specific numbers of ball according to current condition easily.

1. Queues:

* Queue plays very main role in arrangement of Bubbles of specific counts in each row.
* Beauty of using queues is as soon as bottom row of bubble gets destroyed by player queue moves downwards to allows user hassle free gameplay.
* Also, it makes sure that only bubbles which are visible on screen are accessible.

4.***Algorithm***

function initmouse():Integer

i->Input

o->output

ax,bx,cx,dx->registers

1.Start

2.i.x.ax = 0

3.int86(0x33,&i,&o)

4.return(o.x.ax)

function showmouseptr()

1.Start

2.i.x.ax = 1

3.int86(0x33,&i,&o)

4.return

function restrictmouseptr(x1,y1,x2,y2)

1.Start

2.i.x.ax=7

3.i.x.cx=x1

4.i.x.dx=x2

5.int86(0x33,&i,&o)

6.i.x.ax=8

7.i.x.cx=y1

8.i.x.dx=y2

9.int86(0x33,&i,&o)

9.return

function getmousepos(button,x,y)

1.Start

2.i.x.ax = 3

3.int86(0x33,&i,&o)

4.\*button=o.x.bx

5.\*x=o.x.cx

6.\*y=o.x.dx

7.return

function hidemouseptr()

1.Start

2.i.x.ax=2

3.int86(0x33,&i,&o)

4.return

function increase\_level

level->level of bubble network

1.Start

2.level=level+1

3.return

function insertend(t,ele,position)

t ->head pointer

p,q->nodes

ele->element to insert

position->position of element

data->data of node

next->pointer to next node

pos->position of node

start->start of node

1.Start

2.Allocate memory to p of size node

3.p->data=ele

4.p->next=NULL

5. p->pos=position

6. p->l=level

7. if t->start=NULL then

8. t->start=p

9. else

{

10. q=t->start

11. while q->next!=NULL do

12. q=q->next

13. q->next=p

}

14. return

function isempty\_q(ut):Integer

ut->pointer of ultimate\_head

available->Availability of nodes in linked list

ultimate\_start->starting of ultimate\_head

1.Start

2. if ut->ultimate\_start=NULL then

3. return 1

4. return 0

function isfull\_q(ut):Integer

1.Start

2. if ut->available=15 then

3. return 1

4. return 0

function is\_near\_full\_q(ut):Integer

1.Start

2. if ut->available==14 then

3. return 1

4.return 0

function random1():Integer

1.Start

2.Generate random number from 0 to 4 using rand() function and store it in i

3.if i=0 then

4. return 9

5.else if i=1 then

6. return 10

7. else if i=2 then

8. return 11

9. else if i=3 then

10. return 12

11. else if i=4 then

12. return 13

13. else return 0

function turn\_bright(colour):Integer

colour->code number for colour

1.Start

2.if colour=9 then

3. return 1

4. else if colour=10 then

5. return 2

6. else if colour=12 then

7. return 4

8. else if colour=13 then

9. return 5

10. else return 0

function turn\_bright(colour):Integer

colour->code number for colour

1.Start

2.if colour=9 then

3. return 1

4.else if colour=10 then

5. return 2

6. else if colour=12 then

7. return 4

8. else if colour=13 then

9. return 5

10. else return colour

function turn\_light(colour):Integer

1.Start

2.if colour=1 then

3. return 9

4.else if colour=2 then

5. return 10

6. else if colour=4 then

7. return 12

8. else if colour=5 then

9. return 13

10. else return colour

function insert(t)

1.Start

2.increase\_level()

3. if level%2=1 then

4. for i=0 upto i=14 do

5. insertend(t,random1(),i+1)

6. else

7. for i=0 upto i=13 do

8. insertend(t,random1(),i+1)

9.return;

function connect\_left\_right(t)

n->node

left->left of node

right->right of node

1.Start

2. allocate memory to n using malloc()

3.n->data=0

4.n->pos=0

5.n->l=0

6.p=t->start

7.q=t->start

8.p->left=n

9.p->right=p->next

10.p=p->next

11.while p->next!=NULL do

{

12. p->left=q

13. p->right=p->next

14. p=p->next

15. q=q->next

}

16. p->right=n

17. p->left=q

18. return

function connect\_bottom(ut)

hp->head pointer

1.Start

2.allocate memory to n using malloc()

3. n->data=0

4.n->pos=0

5.n->l=0

6.if ut->avaiable!=1 then

7. return

8.hp=ut->ultimate\_start

9. p=hp->start

10.p->left=p->down\_left=p->down\_right=n

11.p->right=p->next

12.p=p->next

13.q=hp->start

14.while p->next!=NULL do

{

15. p->left=q

16. p->right=p->next

17. p->down\_left=p->down\_right=n

18. p=p->next

19. q=q->next

}

20. p->right=n

21. p->down\_left=n

22. p->down\_right=n

23. p->left=q

24. return

function connect(ut)

hq->head pointer

pt,qt,pb,ab->nodes

1.Start

2.allocate memory to n using malloc()

3.n->data=0

4.n->l=0

5.n->pos=0

6.hp=ut->ultimate\_start

7. hq=ut->ultimate\_start

8.while hp->next!=NULL do

9. hp=hp->next

10. while hq->next->next!=NULL do

11. hq=hq->next

12.pt=hp->start

13.qt=hp->start

14.pb=hq->start

15.qb=hq->start

16.if pt->l%2=1 then

{

17. connect\_left\_right(hp)

18. pt->down\_left=n

19. pt->down\_right=pb

20. pb->up\_left=pt

21. pb->up\_right=pt->next

22. pt=pt->next

23. pb=pb->next

24. while pt->next!=NULL do

{

25. pt->down\_left=qb

26. pt->down\_right=pb

27. pb->up\_left=pt

28. pb->up\_right=pt->next

29. if(pb->next!=NULL)

30. pb=pb->next

31. qb=qb->next

32. qt=qt->next

33. pt=pt->next

}

34. pt->down\_right=n

35. pt->down\_left=qb

36. qb->up\_right=pt

}

37. else

{

38. connect\_left\_right(hp)

39. pt->down\_left=pb

40. pt->down\_right=pb->next

41. pb->up\_left=n

42. pb->up\_right=pt

43. pt=pt->next

44. pb=pb->next

45. while pb->next!=NULL do

{

46. pt->down\_left=pb

47. pt->down\_right=pb->next

48. pb->up\_right=pt

49. pb->up\_left=qt

50. pt=pt->next

51. qt=qt->next;

52. pb=pb->nex

53. qb=qb->next

}

54. pb->up\_left=qt

55. pb->up\_right=n

}

56. return

function connect\_top(t)

1.Start

2.allocate memory to n using malloc()

3.n->data=0

4.n->pos=0

5.n->l=0

6.connect\_left\_right(t)

7.display\_node(t)

8.p=t->start

9. while p!=NULL do

{

10. p->up\_left=n

11. p->up\_right=n

12. p=p->next

}

13. return

function display\_node(t)

1.Start

2.p=t->start

3.while(p!=NULL)

{

4. p=p->next

}

5.return

function ultimate\_insert(ut)

1.Start

2.if isfull\_q(ut)=1 then

{

3. return

}

4.hp = (head\*)malloc(sizeof(head))

5.hp->start=NULL

6.hp->next=NULL

7.insert(hp)

8.if isempty\_q(ut)=1then

{

9. ut->ultimate\_start=hp

10 ut->avaiable=1

11. connect\_bottom(ut)

12. display\_node(ut->ultimate\_start)

}

13. else if is\_near\_full\_q(ut) then

{

14. hq=ut->ultimate\_start

15. connect\_top(hp)

16. while hq->next!=NULL do

17. hq=hq->next

18. hq->next=hp

19. ut->avaiable=15

20. connect(ut)

}

21. else

22. {

23. hq = ut->ultimate\_start

24. while hq->next!=NULL do

{

25. hq=hq->next

}

26. hq->next=hp

27. display\_node(hp)

28. ut->avaiable=ut->avaiable+1

29. connect(ut)

}

30. return

function make\_sure\_15(ut)

1.Start

2.set time according to current system time using srand(time(NULL))

3. if ut->avaiable=15 then

4. return

5. while ut->avaiable!=15 do

{

6. ultimate\_insert(ut);

}

7. return

fuction display(ut)

1.Start

2.hp=ut->ultimate\_start

3.while hp!=NULL do

{

4. p=hp->start

5. while p!=NULL do

{

6. printf("%d ",p->data)

7. p=p->next

}

8. Move cursor to next line.

9. if hp->start->l%2=1 then

10. print blank space

11. hp=hp->next

}

12. return

function all\_bright(node \*s,ball \*b)

s->node

b->bubble

x,y->coordinates for mouse pointer

score->scoreof the game

up\_left-> upper left of node

up\_right->upper right of node

down\_left->bottom left of node

down\_right->bottom right of node

1.Start

2. s->data=turn\_bright(b->colour)

3. setcolor(BLACK)

4. for j=1 upto j=10 do

{

5. create circle at current mouse coordinates with radius j

6. delay time for 10 milli-seconds using delay()

}

7. score=score+1

8. if s->right->data=b->colour then

9. all\_bright(s->right,b)

10. if s->left->data=b->colour then

11. all\_bright(s->left,b)

12. if s->up\_right->data=b->colour

13. all\_bright(s->up\_right,b)

14. if s->down\_right->data=b->colour then

15. all\_bright(s->down\_right,b);

16. if s->up\_left->data=b->colour do

17. all\_bright(s->up\_left,b);

18. if s->down\_left->data=b->colour do

19. all\_bright(s->down\_left,b);

20. return

function print\_surrounding(ut,a,p,b)

1.Start

2.hp=ut->ultimate\_start

3.for i=0 upto i=a-1 do

4. hp=hp->next

5. q=hp->start

6. for(i=0;i<p;i++)

7. q=q->next

8. all\_bright(q,b)

9. give\_screen(ut,b)

10.delay time by 100 mill-seconds delay()

11.return

function remove\_all\_bright(ut)

1.Start

2. hp=ut->ultimate\_start

3. while hp!=NULL do

{

4 p=hp->start

5. while p!=NULL do

{

6. if p->data=1 or p->data=2 or p->data=4 or p->data=5

7. p->data=0

8. p=p->next

}

9. hp=hp->next

}

10. return

function check\_for\_all\_zero(ut):Integer

1.Start

2.hp=ut->ultimate\_start

3.p=hp->start

4.while p!=NULL do

{

5. if(p->data!=0)

6. return 0

7. p=p->next

8. }

9. hp->start=NULL

10. hp=hp->next

11. ut->ultimate\_start=ut->ultimate\_start->next

12. ut->avaiable=ut->available-1

13. make\_sure\_15(ut)

14. connect\_bottom(ut)

15. return 1

function loss\_connection(ut)

1.Start

2.hp=ut->ultimate\_start

3.while hp->next!=NULL do

{

4. p=hp->start

5. while p!=NULL do

{

6. if p->up\_left->data=0 and p->up\_right->data=0 and p->data!=0 then

{

7. p->data=0

8. hp=ut->ultimate\_start

}

9. p=p->next

}

10. hp=hp->next

}

11.return

function outer\_rectangle()

1.Start

2.Create an outer rectangle of appropriate size using suitable graphics functions

3.return

function give\_screen(ut,b)

1.Start

2.Create the game screen along with game name “Bubble shooter”, highscore, current score, outer

rectangle, exit, inner rectangle using appropriate graphics functions along with suitable colour

along with network of nodes.

3.return

function exit\_button()

1.Start

2.Create exit button using appropriate graphics functions along with suitable colour in output screen

3.return

Function help\_button()

1.Start

2.Create help button with guidelines for the game using suitable graphics functions.

2.return

function credits\_button()

1.Start

2.Create credit button using appropriate graphics functions along with suitable colour in output

screen

3.return

function credit\_window(m)

1.Start

2.Create credit windows and display contents such as Name and Roll Number of the students using

various graphics functions.

3. return

function opening\_window(m):Integer

1.Start

2.create opening window and display welcome using graphics functions

3.create the outer rectangle using define function outer\_rectangle() and display the game name

“Bubble Shooter” along with buttons such as credits, exit and play along with setting appropriate

Colour

4 maxx=getmaxx()

5. maxy=getmaxy()

6. while kbhit()=false do

{

7. mouse\_click(m)

8. if m->x>=maxx/2-40 and m->y>=maxy-maxy/4-80 and m->x<=maxx/2+40 and m->y<=maxy-

maxy/4 then

{

9. clearviewport()

10. return 1

}

11. if m->x>=10 and m->y>=maxy-30 and m->x<=30 and m->y<=maxy-10 then

{

12. clearviewport()

13. return 2

}

14. if m->x>=maxx-31 and m->x<=maxx-11 and m->y>=maxy-31 and m->y<=maxy-11 then

{

15. clearviewport()

16. credit\_window(m)

17. clearviewport()

18. op=opening\_window(m)

19. return op

}

}

20. getch()

21. clearviewport()

22. return 0

function mouse\_click(m)

1.Start

2. display position of mouse using mouse coordinates m->x and m->y using various graphical

functions

3.return

function find\_touch(ut,b):Integer

1.Start

2.hp=ut->ultimate\_start

3.a=0

4.while hp!=NULL do

{

5. p=hp->start

6. q=0

7. while p!=NULL do

{

8. if (p->data!=0 and b->y<=280 and (p->up\_left->data=0 or p->up\_right->data=0 or p->left-

>data=0 or p->right->data=0 or p->down\_left->data=0 or p->down\_right->data=0))

{

9. d= square root of b->x-p->x raised to 2 +b->y-p->y raised to 2

10. if d<=20 then

{

11. if b->colour= turn\_light(p->data) or b->colour=turn\_bright(p->data) then

{

12. print\_surrounding(ut,a,q,b)

13. remove\_all\_bright(ut)

14. loss\_connection(ut)

15. loss\_connection(ut)

16. setcolor(b->colour)

17. return 1

}

18. else

{

19. delay time by 100 milli-seconds using delay()

20. return 1

}

}

}

21. q=q+1

22. p=p->next

}

23. a++;

24. hp=hp->next

}

25. return 0

function touch(ut,m,b)

m->mouse pointer

1.Start

2.reaction=0

3. create a character pointer buff

4. maxx=getmaxx()

5. maxy=getmaxy()

6. b->x=floating value of maxx/2

7. b->y=floating value of maxy-40

8. set colour to b->colour

9. create circle at current mouse coordinates with radius j

10. area=imagesize((int)b->x-10,(int)b->y-10,(int)b->x+10,(int)b->y+10);

11. allocate memory to buff using malloc() of size area

12. set fill style to font “SOLID\_FILL” and colour ”b->colour” using setfillstyle()

13. floodfill(b->x,b->y,b->colour)

14. getimage((int)b->x-10,(int)b->y-10,(int)b->x+10,(int)b->y+10,buff)

15. x\_move= floating value of m->x-b->x

16. y\_move= floating value b->y-m->y

17. x\_move=x\_move/y\_move

18. y\_move=-1

19. while true do

{

20. if kbhit()=true then

{

21. ch=getch()

22. if ch ='\r' then

{

23. x\_move=-1\*x\_move

24. y\_move=-1\*y\_move

}

25. else

{

26. if ch=27 then

27. break

}

}

28. putimage(integer value of b->x-10,integer value of b->y-10,buff,XOR\_PUT)

29. b->x=b->x+(x\_move)\*1

30. b->y=b->y+(y\_move)\*1

31. putimage(integer value of b->x-10,integer value of b->y-10,buff,XOR\_PUT)

32. reaction=find\_touch(ut,b)

33. if reaction=1 then

34. break

35. delay time by 2 milli-seconds using delay()

36. if b->x>maxx-131-13 or b->x<130+13 then

37. x\_move=-1\*x\_move

38. if b->y>maxy-13 or b->y<57+13 then

39. y\_move=-1\*y\_move

}

40. putimage(integer value of b->x-10,integer value of b->y-10,buff,XOR\_PUT)

41. return

function check\_for\_5\_steps():Integer

1.Start

2 score=score+1

3 if condition=5 then

{

4. current\_state=current\_state+1

5. condition=0

6. return 1

}

7. return 0

function ground\_touch(ut):Integer

1.Start

2.hp=ut->ultimate\_start

3.p=hp->start

4.while p!=NULL do

{

5. if p->y>=270 then

6. return 1

7. p=p->next

}

8. return 0

**5.Implementation: -**

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

#include<stdlib.h>

#include<math.h>

#include"mymouse.h"

typedef struct node

{

int data,pos,l,x,y;

struct node \*next,\*left,\*right,\*up\_left,\*up\_right,\*down\_left,\*down\_right;

}node;

typedef struct head

{

node \*start;

struct head \*next;

}head;

typedef struct ball

{

float x,y;

int colour;

}ball;

typedef struct mouse

{

int button,x,y;

}mouse;

typedef struct ultimte\_head

{

head \*ultimate\_start;

int avaiable;

}ultimate\_head;

long int level,current\_state,condition,score;

int highscore,current\_ball[3];

void display\_node(head \*t);

void increase\_level()

{

level=level+1;

return;

}

int isempty\_q(ultimate\_head \*ut)

{

if(ut->ultimate\_start==NULL)

return 1;

return 0;

}

int isfull\_q(ultimate\_head \*ut)

{

if(ut->avaiable==15)

return 1;

return 0;

}

int is\_near\_full\_q(ultimate\_head \*ut)

{

if(ut->avaiable==14)

return 1;

return 0;

}

int random1()

{

int i=rand()%4;

switch(i)

{

case 0:return 9;

case 1:return 10;

case 2:return 12;

case 3:return 13;

}

return 0;

}

int turn\_bright(int colour)

{

switch(colour)

{

case 9:return 1;

case 10:return 2;

case 12:return 4;

case 13:return 5;

}

return colour;

}

int turn\_light(int colour)

{

switch(colour)

{

case 1: return 9;

case 2: return 10;

case 4: return 12;

case 5:return 13;

}

return colour;

}

int current\_ball\_colour()

{

int colour=current\_ball[0];

current\_ball[0]=current\_ball[1];

current\_ball[1]=current\_ball[2];

current\_ball[2]=random1();

return colour;

}

void insertend(head \*t,int ele,int position)

{

node \*p,\*q;

p=(node\*)malloc(sizeof(node));

p->data=ele;

p->next=NULL;

p->pos=position;

p->l=level;

if(t->start==NULL)

t->start=p;

else

{

q=t->start;

while(q->next!=NULL)

q=q->next;

q->next=p;

}

return;

}

void insert(head \*t)

{

int i;

increase\_level();

if(level%2==1)

for(i=0;i<15;i++)

insertend(t,random1(),i+1);

else

for(i=0;i<14;i++)

insertend(t,random1(),i+1);

return;

}

void connect\_left\_right(head \*t)

{

node \*p,\*q,\*n;

n=(node\*)malloc(sizeof(node));

n->data=0;

n->pos=0;

n->l=0;

p=t->start;

q=t->start;

p->left=n;

p->right=p->next;

p=p->next;

while(p->next!=NULL)

{

p->left=q;

p->right=p->next;

p=p->next;

q=q->next;

}

p->right=n;

p->left=q;

return;

}

void connect\_bottom(ultimate\_head \*ut)

{

head \*hp;

node \*p,\*q ,\*n;

n=(node\*)malloc(sizeof(node));

n->data=0;

n->pos=0;

n->l=0;

if(ut->avaiable!=1)

return;

hp=ut->ultimate\_start;

p=hp->start;

p->left=p->down\_left=p->down\_right=n;

p->right=p->next;

p=p->next;

q=hp->start;

while(p->next!=NULL)

{

p->left=q;

p->right=p->next;

p->down\_left=p->down\_right=n;

p=p->next;

q=q->next;

}

p->right=n;

p->down\_left=n;

p->down\_right=n;

p->left=q;

return;

}

void connect(ultimate\_head \*ut)

{

head \*hp,\*hq;

node \*pt,\*qt,\*pb,\*qb, \*n;

n=(node\*)malloc(sizeof(node));

n->data=0;

n->l=0;

n->pos=0;

hp=ut->ultimate\_start;

hq=ut->ultimate\_start;

while(hp->next!=NULL)

hp=hp->next;

while(hq->next->next!=NULL)

hq=hq->next;

pt=hp->start;

qt=hp->start;

pb=hq->start;

qb=hq->start;

if(pt->l%2==1) //20

{

connect\_left\_right(hp);

pt->down\_left=n;

pt->down\_right=pb;

pb->up\_left=pt;

pb->up\_right=pt->next;

pt=pt->next;

pb=pb->next;

while(pt->next!=NULL)

{

pt->down\_left=qb;

pt->down\_right=pb;

pb->up\_left=pt;

pb->up\_right=pt->next;

if(pb->next!=NULL)

pb=pb->next;

qb=qb->next;

qt=qt->next;

pt=pt->next;

}

pt->down\_right=n;

pt->down\_left=qb;

qb->up\_right=pt;

}

else

{

connect\_left\_right(hp);

pt->down\_left=pb;

pt->down\_right=pb->next;

pb->up\_left=n;

pb->up\_right=pt;

pt=pt->next;

pb=pb->next;

while(pb->next!=NULL)

{

pt->down\_left=pb;

pt->down\_right=pb->next;

pb->up\_right=pt;

pb->up\_left=qt;

pt=pt->next;

qt=qt->next;

pb=pb->next;

qb=qb->next;

}

pb->up\_left=qt;

pb->up\_right=n;

}

return;

}

void connect\_top(head \*t)

{

node \*p,\*n;

n=(node\*)malloc(sizeof(node));

n->data=0;

n->pos=0;

n->l=0;

connect\_left\_right(t);

display\_node(t);

p=t->start;

while(p!=NULL)

{

p->up\_left=n;

p->up\_right=n;

p=p->next;///////////////////////////////////////////forgot

}

return;

}

void display\_node(head \*t)

{

node \*p;

p=t->start;

while(p!=NULL)

{

p=p->next;

}

return;

}

void ultimate\_insert(ultimate\_head \*ut)

{

head \*hp,\*hq;

if(isfull\_q(ut))

{

//printf("error");

return;

}

hp = (head\*)malloc(sizeof(head));

hp->start=NULL;

hp->next=NULL;

insert(hp);

if(isempty\_q(ut))

{

ut->ultimate\_start=hp;

ut->avaiable=1;

connect\_bottom(ut);

//printf("connected bottom sucessfully\n");

display\_node(ut->ultimate\_start);

}

else if(is\_near\_full\_q(ut))

{

//printf("now hwere it's started working");

hq=ut->ultimate\_start;

connect\_top(hp);

//printf("connect top sucessfully");

while(hq->next!=NULL)

hq=hq->next;

hq->next=hp;

ut->avaiable=15;

connect(ut);

}

else

{

//printf("it's running now");

hq = ut->ultimate\_start;

while(hq->next!=NULL)

{

hq=hq->next;

//printf("it goimg next");

}

hq->next=hp;

display\_node(hp);

ut->avaiable=ut->avaiable+1;

connect(ut);

// printf("connected sucessfully ");

}

return;

}

void make\_sure\_15(ultimate\_head \*ut)

{

srand(time(NULL));

if(ut->avaiable==15)

return;

while(ut->avaiable!=15)

{

//printf("making");

ultimate\_insert(ut);

//printf("making done %d",ut->avaiable);

}

return;

}

void display(ultimate\_head \*ut)

{

head \*hp;

node \*p;

hp=ut->ultimate\_start;

while(hp!=NULL)

{

p=hp->start;

while(p!=NULL)

{

printf("%d ",p->data);

p=p->next;

}

printf("\n");

if(hp->start->l%2==1)

printf(" ");

hp=hp->next;

}

return;

}

void give\_screen(ultimate\_head \*ut,ball \*b);

void all\_bright(node \*s,ball \*b)

{

int j;

s->data=turn\_bright(b->colour);

setcolor(BLACK);

for(j=1;j<=10;j++)

{

circle(s->x,s->y,j);

delay(10);

}

score++;

if(s->right->data==b->colour)

all\_bright(s->right,b);

if(s->left->data==b->colour)

all\_bright(s->left,b);

if(s->up\_right->data==b->colour)

all\_bright(s->up\_right,b);

if(s->down\_right->data==b->colour)

all\_bright(s->down\_right,b);

if(s->up\_left->data==b->colour)

all\_bright(s->up\_left,b);

if(s->down\_left->data==b->colour)

all\_bright(s->down\_left,b);

return;

}

void print\_surrounding(ultimate\_head \*ut,int a,int p,ball \*b)

{

head \*hp;

node \*q;

int i;

hp=ut->ultimate\_start;

for(i=0;i<a;i++)

hp=hp->next;

q=hp->start;

for(i=0;i<p;i++)

q=q->next;

all\_bright(q,b);

give\_screen(ut,b);

delay(100);

return;

}

void remove\_all\_bright(ultimate\_head \*ut)

{

head \*hp;

node \*p;

hp=ut->ultimate\_start;

while(hp!=NULL)

{

p=hp->start;

while(p!=NULL)

{

if(p->data==1||p->data==2||p->data==4||p->data==5)

p->data=0;

p=p->next;

}

hp=hp->next;

}

return;

}

int check\_for\_all\_zero(ultimate\_head \*ut)

{

head \*hp;

node \*p;

hp=ut->ultimate\_start;

p=hp->start;

while(p!=NULL)

{

if(p->data!=0)

return 0;

p=p->next;

}

hp->start=NULL;

hp=hp->next;

ut->ultimate\_start=ut->ultimate\_start->next;

ut->avaiable--;

make\_sure\_15(ut);

connect\_bottom(ut);

return 1;

}

void loss\_connection(ultimate\_head \*ut)

{

head \*hp;

node \*p;

hp=ut->ultimate\_start;

while(hp->next!=NULL)

{

p=hp->start;

while(p!=NULL)

{

if(p->up\_left->data==0&&p->up\_right->data==0&&p->data!=0)

{

p->data=0;

hp=ut->ultimate\_start;

}

p=p->next;

}

hp=hp->next;

}

return;

}

void mouse\_click(mouse \*m);

void outer\_rectangle()

{

int maxx=getmaxx();

int maxy=getmaxy();

line(10,0,maxx-11,0); //top

line(0,10,0,maxy-11); //left

line(maxx-1,10,maxx-1,maxy-11); //right

line(10,maxy-1,maxx-11,maxy-1); //bottom line

arc(10,10,90,180,10);

arc(maxx-11,10,0,90,10);

arc(10,maxy-11,180,270,10);

arc(maxx-11,maxy-11,270,0,10);

}

void show\_balls()

{

int i;

int maxy=getmaxy();

for(i=1;i<=3;i++)

{

setcolor(current\_ball[3-i]);

circle(500-i\*30,maxy-30,10);

setfillstyle(SOLID\_FILL,current\_ball[3-i]);

floodfill(500-i\*30,maxy-30,current\_ball[3-i]);

}

return;

}

void help\_menu()

{

int maxx=getmaxx(),ch;

outer\_rectangle();

settextstyle(3,HORIZ\_DIR,7);

outtextxy(maxx/2,40,"HELP");

gotoxy(1,10);

printf(" 1:In this Bubble Shooter you are only allow to touch only single ball.\n");

printf("\n 2:Once ball get touch it will vanished weather it is touch our not.\n");

printf("\n 3:Balls will move down on after every 5 steps.\n");

printf("\n 4:You can change the colour of ball just by simply pressing enter.\n");

printf("\n 5:You can exit the game just by simply pressing esc from keyboard.\n");

printf("\n SO NOW ENJOY\n");

fflush(stdin);

while(!kbhit())

{

ch=getch();

if(ch==27)

{

clearviewport();

return;

}

}

}

void give\_screen(ultimate\_head \*ut,ball \*b)

{

int x1,y1,maxx,state,maxy;

char a[50];

head \*hp;

node \*p;

maxx=getmaxx();

maxy=getmaxy();

hp=ut->ultimate\_start;

y1=current\_state\*18+57;

setcolor(b->colour);

line(130,280,maxx-131,280); //limit line

line(130,67,130,maxy-11); //left limit

line(maxx-131,67,maxx-131,maxy-11); //right limit

arc(140,67,90,180,10); //

arc(maxx-141,67,0,90,10); //

arc(140,maxy-11,180,270,10); //

arc(maxx-141,maxy-11,270,0,10); //for inner rectangle

outer\_rectangle();

line(10,57,maxx-11,57);

arc(10,67,90,180,10);

arc(maxx-11,67,0,90,10);

//score card

line(maxx-111,67,maxx-21,67);

line(maxx-111,67+90,maxx-21,67+90);

line(maxx-121,77,maxx-121,67+80);

line(maxx-11,77,maxx-11,67+80);

arc(maxx-111,77,90,180,10);

arc(maxx-111,67+80,180,270,10);

arc(maxx-21,77,0,90,10);

arc(maxx-21,67+80,270,0,10);

//score inner card

line(maxx-101,107,maxx-31,107);

line(maxx-101,67+80,maxx-31,67+80);

line(maxx-111,117,maxx-111,67+70);

line(maxx-21,117,maxx-21,67+70);

arc(maxx-101,117,90,180,10);

arc(maxx-101,67+70,180,270,10);

arc(maxx-31,117,0,90,10);

arc(maxx-31,67+70,270,0,10);

sprintf(a,"%ld",score);

settextjustify(1,1);

settextstyle(3,HORIZ\_DIR,3);

outtextxy(maxx-66,83,"SCORE");

outtextxy(maxx-66,122,a);

//TILTE

settextstyle(3,HORIZ\_DIR,4);

outtextxy(maxx/2,20,"BUBBLE SHOOTER");

//highscore

line(20,67,100,67);

line(20,157,100,157);

line(10,77,10,147);

line(110,77,110,147);

arc(20,77,90,180,10);

arc(20,147,180,270,10);

arc(100,77,0,90,10);

arc(100,147,270,0,10);

line(30,107,90,107);

line(30,67+80,90,67+80);

line(20,117,20,67+70);

line(100,117,100,67+70);

arc(30,117,90,180,10);

arc(30,67+70,180,270,10);

arc(90,117,0,90,10);

arc(90,67+70,270,0,10);

settextstyle(3,HORIZ\_DIR,1);

outtextxy(62,82,"HIGHSCORE");

if(score>highscore)

highscore=score;

sprintf(a,"%d",highscore);

settextstyle(3,HORIZ\_DIR,3);

outtextxy(60,123,a);

state=0;

// printf("%d",hp->start->l);

while(hp!=NULL)

{

if(state==current\_state)

break;

p=hp->start;

if(p->l%2==1)

x1=maxx/2-165+11;

else

x1=maxx/2-165+11+11;

while(p!=NULL)

{

p->x=x1;

p->y=y1;

setfillstyle(SOLID\_FILL,p->data);

setcolor(p->data);

circle(p->x,p->y,10);

floodfill(p->x,p->y,p->data);

if(p->data==0)

{

setcolor(BLACK);

circle(p->x,p->y,10);

}

p=p->next;

x1=x1+22;

}

y1=y1-19;

hp=hp->next;

state++;

}

return;

}

void play\_button()

{

int maxx=getmaxx();

int maxy=getmaxy();

// rectangle(maxx/2-40,maxy-maxy/4-80,maxx/2+40,maxy-maxy/4);

setcolor(YELLOW);

line(maxx/2-30,maxy-maxy/4-80,maxx/2+30,maxy-maxy/4-80);

line(maxx/2-30,maxy-maxy/4,maxx/2+30,maxy-maxy/4);

line(maxx/2-40,maxy-maxy/4-70,maxx/2-40,maxy-maxy/4-10);

line(maxx/2+40,maxy-maxy/4-70,maxx/2+40,maxy-maxy/4-10);

arc(maxx/2-30,maxy-maxy/4-70,90,180,10);

arc(maxx/2-30,maxy-maxy/4-10,180,270,10);

arc(maxx/2+30,maxy-maxy/4-70,0,90,10);

arc(maxx/2+30,maxy-maxy/4-10,270,0,10);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(maxx/2,maxy-maxy/4-40,YELLOW);

setcolor(BLUE);

line(maxx/2-15,maxy-maxy/4-60,maxx/2-15,maxy-maxy/4-20);

line(maxx/2-15,maxy-maxy/4-60,maxx/2+20,maxy-maxy/4-40);

line(maxx/2-15,maxy-maxy/4-20,maxx/2+20,maxy-maxy/4-40);

setfillstyle(SOLID\_FILL,BLUE);

floodfill(maxx/2,maxy-maxy/4-40,BLUE);

return;

}

void exit\_button()

{

int maxy=getmaxy();

setcolor(WHITE);

circle(20,maxy-21,10);

settextstyle(3,HORIZ\_DIR,1);

outtextxy(20,maxy-24,"E");

return;

}

void help\_button()

{

int maxy=getmaxy();

setcolor(WHITE);

circle(20,maxy-51,10);

settextstyle(3,HORIZ\_DIR,1);

outtextxy(20,maxy-54,"H");

return;

}

void credits\_button()

{

int maxx=getmaxx();

int maxy=getmaxy();

setcolor(WHITE);

circle(maxx-20,maxy-21,10);

settextstyle(3,HORIZ\_DIR,1);

outtextxy(maxx-21,maxy-24,"C");

return;

}

void credit\_window(mouse \*m)

{

int maxx=getmaxx();

int maxy=getmaxy();

outer\_rectangle();

settextstyle(3,HORIZ\_DIR,5);

setcolor(WHITE);

outtextxy(maxx/2,100,"CREDITS");

settextstyle(3,HORIZ\_DIR,3);

outtextxy(maxx/2,maxy/2-50,"Aaditya Rane");

outtextxy(maxx/2,maxy/2,"16102A0066" );

outtextxy(maxx/2,maxy/2+50,"Navanit Srisangkar");

outtextxy(maxx/2,maxy/2+100,"16102A0065");

outtextxy(maxx/2,maxy/2+150,"Ram Agrawal");

outtextxy(maxx/2,maxy/2+200,"16102A0053");

exit\_button();

while(!kbhit())

{

mouse\_click(m);

if(m->x>=10&&m->y>=maxy-30&&m->x<=30&&m->y<=maxy-10)

{

clearviewport();

return;

}

}

getch();

return;

}

void partial\_opening\_window()

{

int maxx=getmaxx();

int maxy=getmaxy();

clearviewport();

setcolor(15);

outer\_rectangle();

settextstyle(3,HORIZ\_DIR,5);

outtextxy(maxx/2,maxy/6,"{BUBBLE SHOOTER}");

setcolor(13);

line(140,55,480,55);

line(140,120,480,120);

play\_button();

exit\_button();

credits\_button();

help\_button();

return;

}

int opening\_window(mouse \*m)

{

int maxx=getmaxx(),j,op;

int maxy=getmaxy();

setcolor(RED);

settextjustify(1,1);

settextstyle(3,HORIZ\_DIR,10);

for(j=0;j<10;j++)

{

outer\_rectangle();

setcolor(j+1);

settextstyle(1,HORIZ\_DIR,j);

outtextxy(maxx/2,maxy/2,"WELCOME");

delay(100);

clearviewport();

}

outtextxy(maxx/2,maxy/2,"WELCOME");

for(j=2;j<=maxx/2;j++)

{

setcolor(BLACK);

line(j,40,j,maxy-40);

line(maxx-j,40,maxx-j,maxy-40);

delay(5);

}

setcolor(15);

outer\_rectangle();

settextstyle(3,HORIZ\_DIR,5);

outtextxy(maxx/2,maxy/6,"{BUBBLE SHOOTER}");

setcolor(13);

line(140,55,480,55);

line(140,120,480,120);

play\_button();

exit\_button();

credits\_button();

help\_button();

while(!kbhit())

{

mouse\_click(m);

if(m->x>=maxx/2-40&&m->y>=maxy-maxy/4-80&&m->x<=maxx/2+40&&m->y<=maxy-maxy/4)

{

clearviewport();

return 1;

}

if(m->x>=10&&m->y>=maxy-30&&m->x<=30&&m->y<=maxy-10)

{

clearviewport();

return 2;

}

if(m->x>=10&&m->y>=maxy-60&&m->x<=30&&m->y<=maxy-40)

{

clearviewport();

help\_menu();

partial\_opening\_window();

mouse\_click(m);

}

if(m->x>=maxx-31&&m->x<=maxx-11&&m->y>=maxy-31&&m->y<=maxy-11)

{

clearviewport();

credit\_window(m);

clearviewport();

op=opening\_window(m);

return op;

}

}

getch();

clearviewport();

return 0;

}

void mouse\_click(mouse \*m)

{

while(!kbhit())

{

getmousepos(&m->button,&m->x,&m->y);

// gotoxy(5,3);

// (m->button&1)==1?printf("down"):printf("UP");

if(m->button&1==1)

return;

// gotoxy(20,3);

// (m->button&2)==2?printf("DOWN"):printf("UP");

// gotoxy(65,2);

// printf("X=%03d y=%03d",m->x,m->y);

}

return;

}

int find\_touch(ultimate\_head \*ut,ball \*b)

{

int q,a,j;

head \*hp;

node \*p;

float d;

hp=ut->ultimate\_start;

a=0;

while(hp!=NULL)

{

p=hp->start;

q=0;

while(p!=NULL)

{

if(p->data!=0&&b->y<=280&&(p->up\_left->data==0||p->up\_right->data==0||p->left->data==0||p->right->data==0||p->down\_left->data==0||p->down\_right->data==0))

{

d=sqrt(pow(b->x-p->x,2)+pow(b->y-p->y,2));

if(d<=20)

{

if(b->colour==turn\_light(p->data)||b->colour==turn\_bright(p->data))

{

print\_surrounding(ut,a,q,b);

remove\_all\_bright(ut);

loss\_connection(ut);

loss\_connection(ut);

setcolor(b->colour);

return 1;

}

else

{

delay(500);

return 1;

}

}

}

q++;

p=p->next;

}

a++;

hp=hp->next;

}

return 0;

}

void touch(ultimate\_head \*ut,mouse \*m,ball \*b)

{

float x\_move,y\_move;

int maxx,maxy,ch,area,reaction=0;

char \*buff;

maxx=getmaxx();

maxy=getmaxy();

b->x=(float)maxx/2;

b->y=(float)maxy-40;

setcolor(b->colour);

circle(b->x,b->y,10);

area=imagesize((int)b->x-10,(int)b->y-10,(int)b->x+10,(int)b->y+10);

buff=(char\*)malloc(area);

setfillstyle(SOLID\_FILL,b->colour);

floodfill(b->x,b->y,b->colour);

getimage((int)b->x-10,(int)b->y-10,(int)b->x+10,(int)b->y+10,buff);

x\_move=(float)m->x-b->x;

y\_move=(float)b->y-m->y;

x\_move=x\_move/y\_move;

y\_move=-1;

//getch();

while(1)

{

if(kbhit())

{

ch=getch();

if(ch=='\r')

{

x\_move=-1\*x\_move;

y\_move=-1\*y\_move;

}

else

{

if(ch==27)

break;

}

}

putimage((int)b->x-10,(int)b->y-10,buff,XOR\_PUT);

b->x=b->x+(x\_move)\*1;

b->y=b->y+(y\_move)\*1;

putimage((int)b->x-10,(int)b->y-10,buff,XOR\_PUT);

reaction=find\_touch(ut,b);

if(reaction==1)

break;

delay(2);

if(b->x>maxx-131-13||b->x<130+13)

x\_move=-1\*x\_move;

if(b->y>maxy-13||b->y<57+13)

y\_move=-1\*y\_move;

}

putimage((int)b->x-10,(int)b->y-10,buff,XOR\_PUT);

return;

}

int check\_for\_5\_steps()

{

score++;

if(condition==5)

{

current\_state++;

condition=0;

return 1;

}

else

return 0;

}

int ground\_touch(ultimate\_head \*ut)

{

head \*hp;

node \*p;

hp=ut->ultimate\_start;

p=hp->start;

while(p!=NULL)

{

if(p->y>=270)

return 1;

p=p->next;

}

return 0;

}

int main()

{

int a,p,ch,gd=DETECT,gm,maxx,maxy,op\_ch,i;

FILE \*fptr;

mouse m;

ball b;

ultimate\_head u;

u.ultimate\_start=NULL;

u.avaiable=0;

level=0;

condition=0;

current\_state=7;

score=0;

fptr=fopen("C:\\project.txt","r");

fscanf(fptr,"%d",&highscore);

printf("%d",highscore);

fclose(fptr);

for(i=0;i<3;i++)

current\_ball[i]=random1();

initgraph(&gd,&gm,"C:\\TurboC3\\BGI");

initmouse();

showmouseptr();

maxx=getmaxx();

maxy=getmaxy();

make\_sure\_15(&u);

srand(time(NULL));

op\_ch=opening\_window(&m);

if(op\_ch==2)

return 0;

hidemouseptr();

clearviewport();

showmouseptr();

restrictmouseptr(0,0,maxx-1,380);

while(1)

{

if(kbhit())

{

ch=getch();

if(ch==27)

break;

}

fflush(stdin);

b.colour=current\_ball\_colour();

setcolor(b.colour);

circle(maxx/2,maxy-40,10);

setfillstyle(SOLID\_FILL,b.colour);

floodfill(maxx/2,maxy-40,b.colour);

give\_screen(&u,&b);

show\_balls();

mouse\_click(&m);

hidemouseptr();

if(m.button&1==1)

{

setfillstyle(SOLID\_FILL,BLACK);

floodfill(maxx/2,maxy-40,b.colour);

setcolor(BLACK);

circle(maxx/2,maxy-40,10);

setcolor(WHITE);

touch(&u,&m,&b);

condition++;

if(check\_for\_5\_steps())

clearviewport();

setfillstyle(SOLID\_FILL,BLACK);

bar(maxx-101,117,maxx-31,67+70);

give\_screen(&u,&b);

while(1)

{

if(check\_for\_all\_zero(&u)==0)

break;

else

{

if(current\_state>=2)

current\_state--;

clearviewport();

}

}

}

showmouseptr();

if(ground\_touch(&u))

break;

}

if(score>=highscore)

{

fptr=fopen("C:\\project.txt","w");

fprintf(fptr,"%d",score);

fclose(fptr);

}

return 0;

}

6.RESULTS AND CONCLUSION:

* This concludes that just by simple use of linked list with help of some complex logic one can create beautiful games like Bubble shooter.
* Just little beautiful touch of graphics makes this game user friendly.
* Also, use of mouse function allows user to interact with game.
* This project is representation of how one can use data structure in problem based applications.
* Somewhat similar logic can be used in making of various board games like chess, Ludo, tic-tac-toe, snakes and ladders, checkers since every above game contains interconnected boxes inside it.
* Also, one can extend this game with including levels, some better use of files allows one to even interconnect this game with his friends and to allow them to play it on their pc and use single online uploaded file to maintain everyone’s score.

**Screenshots of output screen:**

 

 

 

