**Balloon Burst**

**A Graphics Game Project**

**Made by:**

**Rishav Anand, Rishi Padhi,**

**Ritesh Upadhyay & Shubham Rastogi**

**November 2012**

**Abstract**

The game is fairly simple with the objective of popping as many balloons as possible with the most accurate clicks. The game calculates the no. of clicks, the no. of misses and the no. of hits. It is small yet very addictive game and is made using C graphics library.

Contents:

* Certificates
* Designing the GUI
* Creating the object for the game
* Programming the game
* Objective and how to play
* Bugs that were encountered
* Bibliography
* Conclusion

Certificate

This is to certify that **Rishav Anand** has taken part and made a contribution in the making of the graphics game project **“Balloon Burst”** during the session 2012-13 in **KIIT University** as a part of his course in **Computer Graphics and Multimedia** along with his team members Rishi Padhi, Ritesh Upadhyay & Shubham Rastogi under the guidance of

**Prof. Srinadh Babu N**

Signature University Signature

of Professor Seal of the Dean

Date:

Certificate

This is to certify that **Rishi Padhi** has taken part and made a contribution in the making of the graphics game project **“Balloon Burst”** during the session 2012-13 in

**KIIT University** as a part of his course in **Computer Graphics and Multimedia** along with his team members Rishav Anand, Ritesh Upadhyay & Shubham Rastogi under the guidance of

**Prof. Srinadh Babu N**

Signature University Signature

Of Professor Seal of the Dean

Date:

Certificate

This is to certify that **Ritesh Upadhyay** has taken part and made a contribution in the making of the graphics game project **“Balloon Burst”** during the session

2012-13 in **KIIT University** as a part of his course in **Computer Graphics and Multimedia** along with his team members Rishav Anand, Rishi Padhi & Shubham Rastogi under the guidance of

**Prof. Srinadh Babu N**

Signature University Signature

Of Professor Seal of the Dean

Date:

Certificate

This is to certify that **Shubham Rastogi** has taken part and made a contribution in the making of the graphics game project **“Balloon Burst”** during the session

2012-13 in **KIIT University** as a part of his course in **Computer Graphics and Multimedia** along with his team members Rishav Anand, Rishi Padhi & Ritesh Upadhyay under the guidance of

**Prof. Srinadh Babu N**

Signature University Signature

of Professor Seal of the Dean

Date:

**OBJECT FOR THE GAME**

This game required one object i.e. balloon. This object is made out of following algorithms:

1. Bresenham’s Circle Drawing Algorithm
2. Mid-point Ellipse Drawing Algorithm

And some predefined computer graphics functions as:

1. Arc Function [void arc (int x, int y, int stangle, int endangle, int radius);]
2. Line Function [void line(int x1, int y1, int x2, int y2);]

The following steps were performed to make the balloon:

* The top part of the balloon was made using Bresenham’s Circle Drawing Algorithm and the bottom part was made using Mid-point Ellipse Drawing Algorithm keeping the center of both the circle and ellipse the same.
* The radius of the circle is equal to the minor axis [a] of the ellipse and the major axis [b] was adjusted accordingly to give the figure the shape of a balloon.
* The tail end of the balloon was made using arc and line function of graphics library in C.

**DESIGNING THE GUI**

The GUI of the game area has been made to look like the background of one of the stalls in carnival/festival where people use darts to pop balloons. On the lower part of the screen the scores are displayed with boxes to represent the no. of shots, misses and bursts.

The functions implemented in creating the background are:

* rectangle
* line
* arc
* outtextxy
* floodfill
* setfillpattern
* setcolor
* cleardevice

The description of the background is:

* The game area is made using the setfillpattern function to create a mesh like background of grey and black
* The top and the bottom arcs have been decorated with red colour as like the object (balloon).
* The upper part of the stage above which the objects are generated is filled with grey.
* The rest of the objects are filled with floodfill and setfillpattern to decorate the stage.

The text used to represent the scores is displayed using outtextxy function with a programmable counter which updates the score based on the game rules.

cleardevice function is used to refresh the screen so that the every time the object is translated it appears to be moving smoothly without leaving a trail.

**PROGRAMMING THE GAME**

The following functions were used while making the game:

1. **Balloon Function:**

This function was used to create the balloon (object).

Parameters used are:

* ‘a’ -> X co-ordinate for centre
* ‘b’ -> Y co-ordinate for centre
* ‘r’ -> Radius of the balloon
* ‘e’ -> major axis of ellipse, which comprises of the lower part of balloon.

1. **Increase\_shots Function:**

This function was used to increment the counter “counter 2”, whenever the player attempts to hit the balloon (It doesn’t matter whether it’s a hit or miss).

It updates the no. of attempts made by the player.

No parameters were used in this function.

1. **Increase\_burst Function:**

This function was used to increment the counter “counter 1”, whenever the player attempts to hit the balloon and they actually hit it (by clicking the mouse).It updates the no. of burst made by the player.

No parameter were used in this function.

1. **Increase\_misses Function:**

This function was used to increment the counter “counter 3”, whenever the player attempts to hit the balloon and they cannot hit it (by clicking the mouse).

It updates the no. of misses made by the player.

No parameter were used in this function.

1. **Background Function:**

This function is used to design the background of the game. No parameters were used in this function.

1. **Click\_handler Function:**

This function is called whenever the mouse is clicked on the game screen. And if the click is made on the red colour of the balloon, then the Boolean variable red\_clicked is made true.

This function is used at the beginning of the game and as well as at the end of the game too. And it works just like it would work for the balloon.

1. **Borland Function:**

The whole game is made using Borland Function (Except when we used user defined algorithms like Bresenhem’s algorithms and Mid-point ellipse algorithms).

* **Settextjustify:**

**Syntax:** void settextjustify(int horiz, int vert);

**Description:**

Text output after a call to settextjustify is justified around the current position (CP) horizontally and vertically, as specified. The default justification settings are LEFT\_TEXT (for horizontal) and TOP\_TEXT (for vertical). The enumeration text\_just in graphics.h provides names for the horiz and vert settings passed to settextjustify.

## Settextstyle:

**Syntax:** void settextstyle(int font, int direction, int charsize);

## Description:

## Settextstyle sets the text font, the direction in which text is displayed, and the size of the characters. A call to settextstyle affects all text output by outtext and outtextxy.

## Outtextxy:

**Syntax:** void outtextxy(int x, int y, char \*textstring);

## Description:

Outtextxy displays a text string in the viewport at the given position (x, y), using the current justification settings and the current font, direction, and size.

To maintain code compatibility when using several fonts, use textwidth and textheight to determine the dimensions of the string. If a string is printed with the default font using outtext or outtextxy, any part of the string that extends outside the current viewport is truncated.

## Setcolor:

**Syntax:** void setcolor(int color);

## Description:

## setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor. The current drawing color is the value to which pixels are set when lines, and so on are drawn. The drawing colors shown below are available for the CGA and EGA, respectively.

## Setfillstyle:

**Syntax:** void setfillstyle(int pattern, int color);

## Description:

Setfillstyle sets the current fill pattern and fill color. To set a user-defined fill pattern, do not give a pattern of 12 (USER\_FILL) to setfillstyle; instead, call setfillpattern.

If invalid input is passed to setfillstyle, graphresult returns -1(grError), and the current fill pattern and fill color remain unchanged.

* **Floodfill:**

**Syntax:** void floodfill(int x, int y, int border);

**Description:**

Floodfill fills an enclosed area on bitmap devices. (x,y) is a "seed point" within the enclosed area to be filled. The area bounded by the color border is flooded with the current fill pattern and fill color. If the seed point is within an enclosed area, the inside will be filled. If the seed is outside the enclosed area, the exterior will be filled.

* **Line:**

**Syntax:** void line(int x1, int y1, int x2, int y2);

## Description:

## Line draws a line in the current color, using the current line style and thickness between the two points specified, (x1,y1) and (x2,y2), without updating the current position (CP).

## Arc:

## Syntax: void arc (int x, int y, int stangle, int endangle, int radius);

## Description:

## Arc draws a circular arc in the current drawing color centered at (x,y) with a radius given by radius. the arc travels from stangle to endangle. if stangle equals 0 and endangle equals 360, the call to arc draws a complete circle.

* **Rectangle:**

**Syntax:** void rectangle(int left, int top, int right, int bottom);

**Description:**

Rectangle draws a rectangle in the current line style, thickness, and drawing color. (left,top) is the upper left corner of the rectangle, and (right,bottom) is its lower right corner.

## Registermousehandler:

**Syntax:** void registermousehandler(int kind, void h(int, int));

**Description:**

The registermousehandler function is available in the [winbgim](http://www.cs.colorado.edu/%7Emain/bgi/README.html) implementation of BGI graphics. In general, you write a different "handler function" to handle each different kind of mouse event, and you "register" each of your handlers by calling registermousehandler. The first argument to registermousehandler is one of these constants from the graphics.h file:

WM\_MOUSEMOVE

if you want the handler called whenever the mouse moves

WM\_LBUTTONDBLCLK

...called whenever the left mouse button is double clicked

WM\_LBUTTONDOWN

...called whenever the left mouse button is clicked down

WM\_LBUTTONUP

...called whenever the left mouse button is released up

WM\_MBUTTONDBLCLK

...called whenever the middle mouse button is double clicked

WM\_MBUTTONDOWN

...called whenever the middle mouse button is clicked down

WM\_MBUTTONUP

...called whenever the middle mouse button is released up

WM\_RBUTTONDBLCLK

...called whenever the right mouse button is double clicked

WM\_RBUTTONDOWN

...called whenever the right mouse button is clicked down

WM\_RBUTTONUP

...called whenever the right mouse button is released up

The second argument to registermousehandler must be the name of the handler function that you wrote. This function must be a void function with two int parameters. Whenever the specified mouse event occurs, your handler will be called and the two int parameters will be the x and y positions where the event happened.

* **Initwindow:**

**Syntax:** int initwindow(int width, int height, const char\* title="Windows BGI", int left=0, int top=0, bool dbflag=false, closeflag=true);

**Description:**

The initwindow function is available in the winbgim implementation of BGI graphics. You do not need to include conio.h; just include graphics.h. The function initializes the graphics system by opening a graphics window of the specified size. The first two parameters (width and height) are required, but all other parameters have default values.

The title parameter is the title that will be printed at the top of the window (with a default of "Windows BGI".) If this is set to the empty string (no characters), then the window will be opened without a title bar or border (typically used for a popup message that the user can then close by clicking), and the user will not be able to move this window. If you want a title bar with no visible title, then set the title to a string containing one space.

The left and top parameters determine the screen coordinates of the left and top sides of the window.

The dbflag parameter determines whether double-buffering for the window is automatically turned on as described in the swapbuffers function (true means that double-buffering will be turned on).

If the closeflag parameter is true, then the user can click on the window's close button to shut down the entire program.

## Cleardevice:

**Syntax:** void cleardevice(void);

**Description:**

Cleardevice erases (that is, fills with the current background color) the entire graphics screen and moves the CP (current position) to home (0,0).

* **Putpixel:**

**Syntax:** void putpixel(int x, int y, int color);

**Description:**

Putpixel plots a point in the color defined by color at (x,y).

* **Getpixel:**

**Syntax:** unsigned getpixel(int x, int y);

**Description:**

Getpixel gets the color of the pixel located at (x,y).

* **Delay:**

**Syntax:** void delay(int millisec);

**Description:**

The delay function is available in the winbgim implementation of BGI graphics. You do not need to include conio.h; just include graphics.h. The function pauses the computation for the the specified number of milliseconds.

**OBJECTIVE AND**

**HOW-TO-PLAY**

**Objective:**

It is a fairly simple game, we have to pop the balloons that are floating on the screen. Score would be given on the basis of how many SHOTS actually pop the balloon (we call this as a BURST), if any click does not pop the balloon it would be counted as a MISS.

The ratio of BURSTs to total no. of SHOTS gives the players accuracy.

**How-to-Play:**

The game is played using the Mouse. Every left click made when the pointer is on the game area on screen is counted as a SHOT. The game area is the region where the balloons appear to be floating.

The balloons are of red colour while the background is a black-grey pattern. Every time the user makes a SHOT (left clicks) on the balloon the player gets one point for making a HIT. If the player makes a SHOT and the cursor is not on the balloon but within the game area, the attempt counts as a MISS. The game goes on for fifty balloons. If the user HITs all fifty balloon without a MISS, he has made the highest score with 100% accuracy.

**BIBLIOGRAPHY**

* [www.cs.colorado.edu](http://www.cs.colorado.edu)
* Computer Graphics (C Version) by Donald D. Hearn & M. Pauline Baker
* Wikipedia

**CONCLUSION**

The project was a learning experience as it taught us how we could make a simple game with the basic graphic functions defined in C. It also helped us to understand how game programs work and with the guidance of our mentor and teacher, Prof. Srinadh Babu N. we were able to complete this project successfully.