PROJECT REPORT

GUI :Lights Off Game

Using Gambas3

14 MAY, 2019

Submitted By,

Haseena Hassan (Roll no : 63)
Akshay C Pradeep (Roll no : 62)
Vyshnavi E K(Roll no : 60)

S4 CSE , COLLEGE OF ENGINEERING , TRIVANDRUM

Contents

Aim

To create a GUI project using the free and open source GUI software Gambas

Introduction

Gambas is the name of an object-oriented dialect of the BASIC Programming Language, as well as the integrated development environment that accompanies it. Designed to run on Linux and other Unix-Like computer operating systems its name is a recursive acronym for ***G****ambas* ***A****lmost* ***M****eans* ***Bas****ic*. Gambas is also the word for prawns in the Spanish, French, and Portugese languages, from which the project's logos are derived.

We have made use of Gambas in this project to make a simple retro game . We have learned to used the important features of the game by building this project.

Software Requirements Specifications

1. Linux Os

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. Most Linux distributions, as collections of software based around the Linux kernel and often around a package management system, provide complete LAMP setups through their packages.

2.Gambas 3.x

Gambas is a full-featured object language and development environment built on a BASIC interpreter.
It is released under the GNU General Public Licence.

Its architecture is largely inspired by Java. So Gambas is made up of:

- A compiler.
- An interpreter.
- An archiver.
- A scripter.

- A development environment.
- Many extension components.

The compiler is a fast little executable written in C

Gambas is very useful for developing GUI applications and Games that are good looking and fun to play . Gambas can incorporate other GUI libraries like QT and GTK making it flexible and popular among GUI programmers.

Gambas also has examples and plenty of supporing mateial available online and offline(available with the application) making its learning an enriching experience.

Installation of Gambas

How to install Gambas 3.9.0 in Ubuntu via PPA:

1. Before installing Gambas 3.9.0, remove the previous release if installed:

```
sudo apt remove gambas3 && sudo apt autoremove
```

Type in your password when it asks and hit Enter

2. Add Gambas PPA via command:

```
sudo add-apt-repository ppa:gambas-team/gambas3
```

gambas3 ppa

3. Update and install the new Gambas 3.9.0 release via:

```
sudo apt update
```

```
sudo apt install gambas3
```

Revert to original Gambas in Ubuntu:

ppa-purge will not downgrade the software due to a package conflict. So to undo changes:

remove the current release via the command in step 1:

```
sudo apt remove gambas3 && sudo apt autoremove
```

remove the PPA via command:

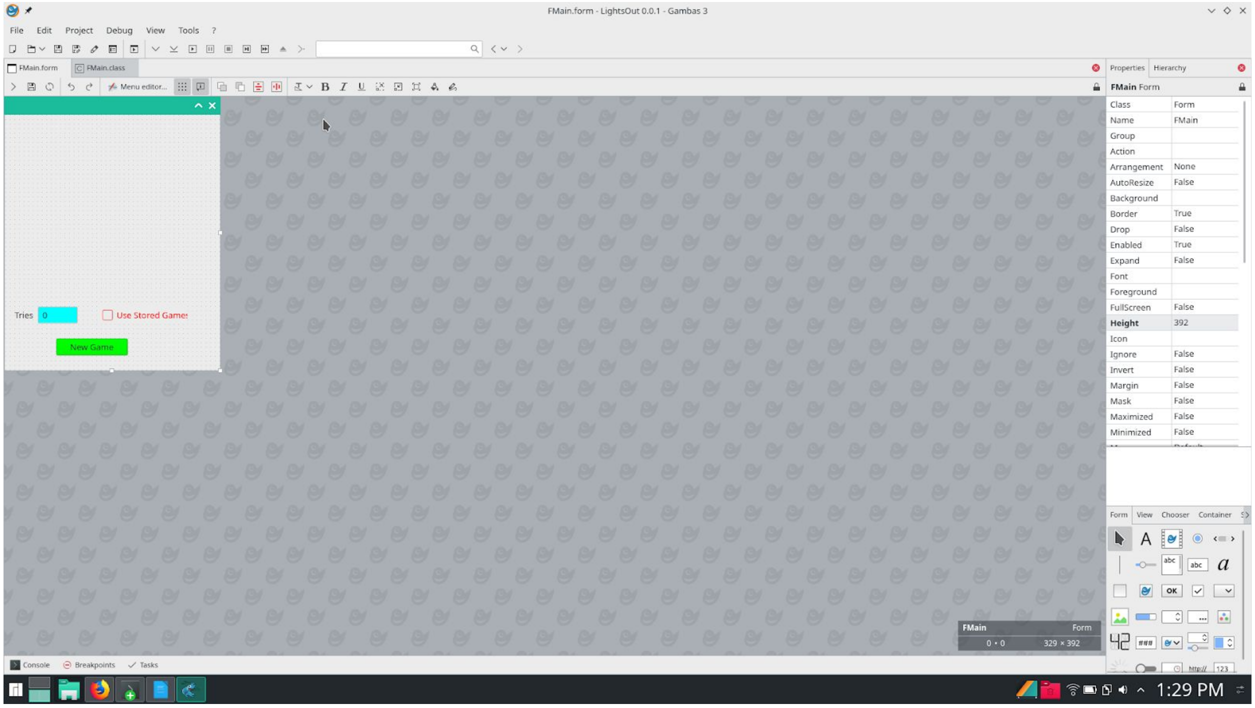
```
sudo add-apt-repository --remove ppa:gambas-team/gambas3
```

finally reinstall the stock version of gambas in Ubuntu:

```
sudo apt update && sudo apt install gambas3
```

If you are using any other distro other than Ubuntu thrn you can use your favourite package manager.
Eg: pacman, octopi or pamac for arch based distros like manjaro .

Game window setting

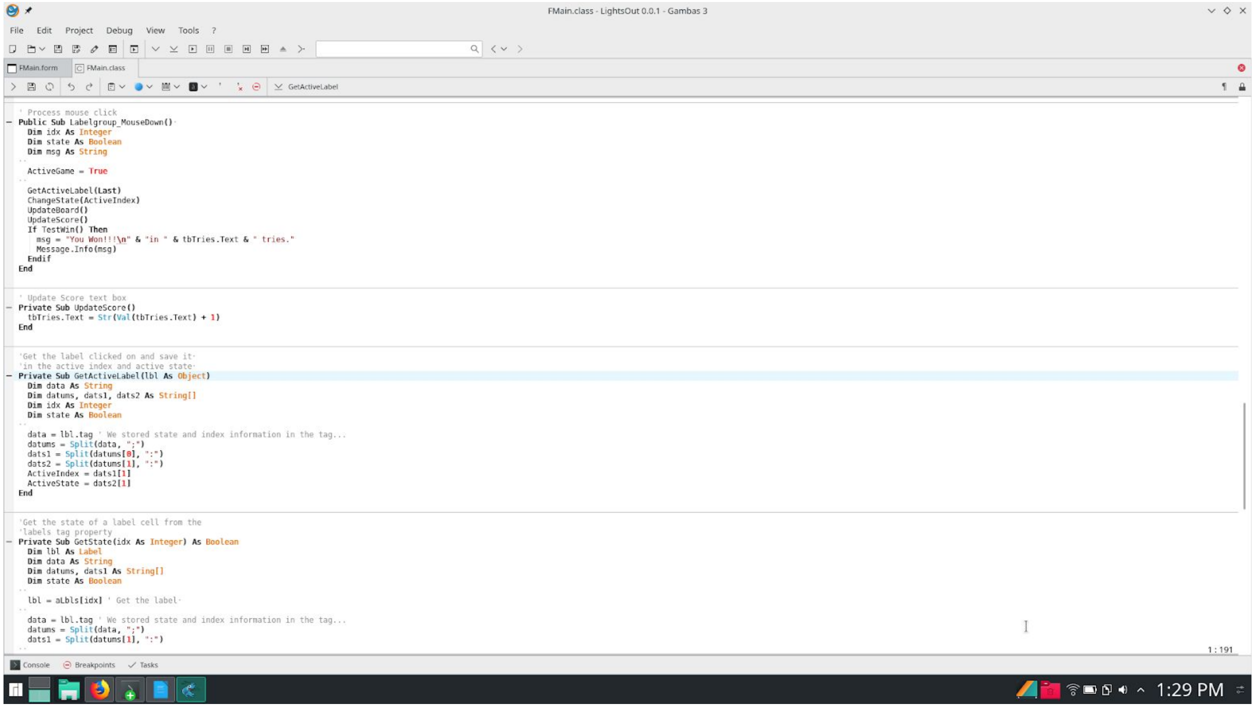


Properties | Hierarchy

FMain Form

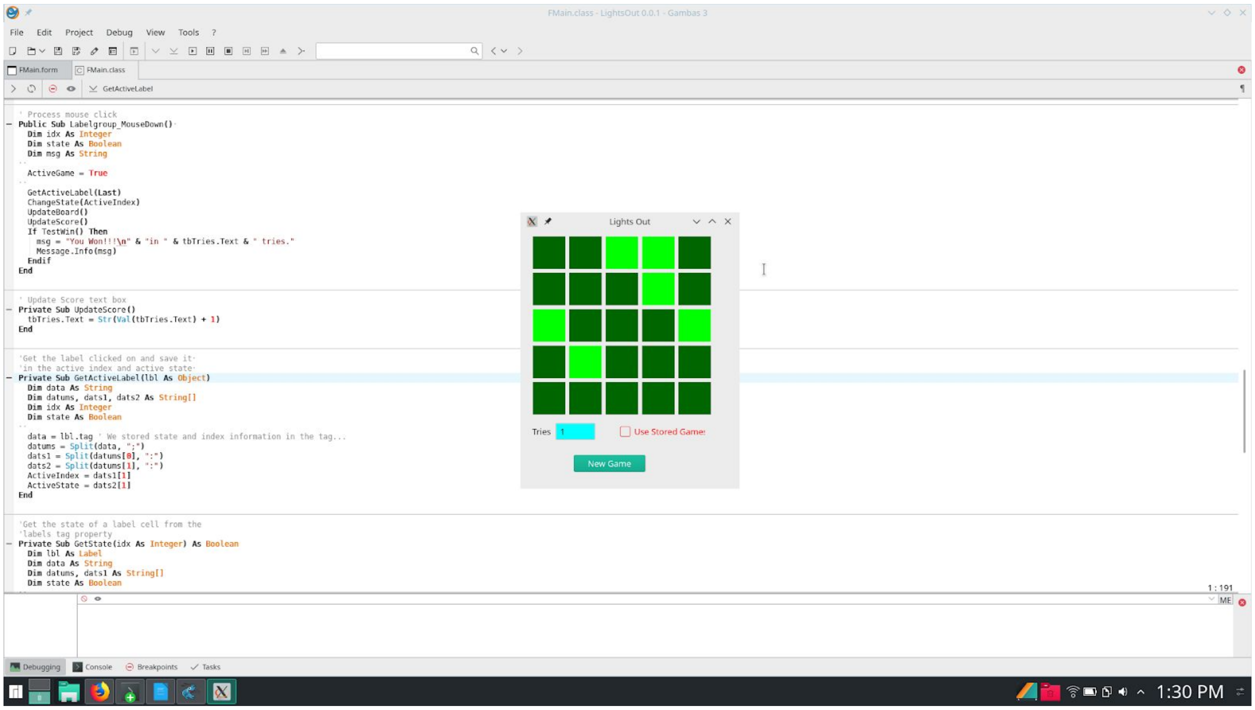
Class	Form
Name	FMain
Group	
Action	
Arrangement	None
AutoResize	False
Background	
Border	True
Drop	False
Enabled	True
Expand	False
Font	
Foreground	
FullScreen	False
Height	392
Icon	
Ignore	False
Invert	False
Margin	False
Mask	False
Maximized	False
Minimized	False

Here you can drag and drop the components like textbox, buttons,images,labels,text and many other items that are available at the bottom left corner. We can add features and properties to all these elements by clicking on them and changing their properties on the right side panel . It include properties like text, name fontcolor, font etc.



```
' Process mouse click
Public Sub Labelgroup_MouseDown()
  Dim idx As Integer
  Dim state As Boolean
  Dim msg As String

  ActiveGame = True

  GetActiveLabel(Last)
  ChangeState(ActiveIndex)
  UpdateBoard()
  UpdateScore()
  If TestWin() Then
    msg = "You Won!!!\n" & "in " & tbTries.Text & " tries."
    Message.Info(msg)
  Endif
End

' Update Score text box
Private Sub UpdateScore()
  tbTries.Text = Str(Val(tbTries.Text) + 1)
End

'Get the label clicked on and save it
'in the active index and active state
Private Sub GetActiveLabel(lbl As Object)
  Dim data As String
  Dim datums, dats1, dats2 As String[]
  Dim idx As Integer
  Dim state As Boolean

  data = lbl.tag ' We stored state and index information in the tag...
  datums = Split(data, ";")
  dats1 = Split(datums[0], ":")
  dats2 = Split(datums[1], ":")
  ActiveIndex = dats1[1]
  ActiveState = dats2[1]
End

'Get the state of a label cell from the
'labels tag property
Private Sub GetState(idx As Integer) As Boolean
  Dim lbl As Label
  Dim data As String
  Dim datums, dats1 As String[]
  Dim state As Boolean

  lbl = aLbls[idx] ' Get the label

  data = lbl.tag ' We stored state and index information in the tag...
  datums = Split(data, ";")
  dats1 = Split(datums[1], ":")
```

On the next page we can add our code. This code is used to adjust timings ,add movement, specify conditions for win, game over, calculate points, count the number of tries etc.



```
' Process mouse click
Public Sub Labelgroup_MouseDown()
  Dim idx As Integer
  Dim state As Boolean
  Dim msg As String

  ActiveGame = True

  GetActiveLabel(Last)
  ChangeState(ActiveIndex)
  UpdateBoard()
  UpdateScore()
  If TestWin() Then
    msg = "You Won!!!\n" & "in " & tbTries.Text & " tries."
    Message.Info(msg)
  Endif
End

' Update Score text box
Private Sub UpdateScore()
  tbTries.Text = Str(Val(tbTries.Text) + 1)
End

'Get the label clicked on and save it
'in the active index and active state
Private Sub GetActiveLabel(lbl As Object)
  Dim data As String
  Dim datums, dats1, dats2 As String[]
  Dim idx As Integer
  Dim state As Boolean

  data = lbl.tag ' We stored state and index information in the tag...
  datums = Split(data, ";")
  dats1 = Split(datums[0], ":")
  dats2 = Split(datums[1], ":")
  ActiveIndex = dats1[1]
  ActiveState = dats2[1]
End

'Get the state of a label cell from the
'labels tag property
Private Sub GetState(idx As Integer) As Boolean
  Dim lbl As Label
  Dim data As String
  Dim datums, dats1 As String[]
  Dim state As Boolean
```

Source Code

```
' Gambas class file


Public iColorFalse As Integer
Public iColorTrue As Integer

Private aLbls As Label[]

Private ActiveIndex As Integer
Private ActiveState As Boolean

Private ActiveGame As Boolean
Private bUseStoredGames As Boolean

Private NumberOfStoredGames As Integer
Private MaxStartingPositions As Integer

Private iRowSize As Integer = 5
Private iColSize As Integer = 5

Public Sub _new()

End


' Set up initial conditions
Public Sub Form_Open()
  aLbls = New Label[(iRowSize * iColSize)]
  ActiveState = Int(False)
  ActiveIndex = 0
  iColorFalse = &H006600
  iColorTrue = &H00FF00
  bUseStoredGames = False
  NumberOfStoredGames = 8
  MaxStartingPositions = 16

  Me.Center
  ResetBoard()
  NewGame()
```

```
End


' Clear board
Private Sub ResetBoard()
  Dim i, cellSize, cellSpace As Integer
  Dim rowOffset, colOffset As Integer
  Dim GridSize As Integer

  ActiveGame = False
  tbTries.Text = "0"
  gridSize = 5 ' Five cells Horz and Vert
  cellSize = 50 ' cell is 50 x 50 px
  cellSpace = 6 '6px between cells
  rowOffset = 8 '8px down from top
  colOffset = 15 '16px right from 0

  'Draw board
  For i = 0 To 24
          aLbls[i] = New Label(Me) As "Labelgroup"    ' Create a new label in the array and store it
in an action group
          With aLbls[i]
          .X = ((i Mod gridSize) * (cellSize + cellSpace)) + colOffset 'Calculate column placement
          .Y = (i \ gridSize) * (cellSize + cellSpace) + rowOffset 'Calculate row placement
          .Width = cellSize
          .Height = cellSize
          .Tracking = True
          .Background = iColorFalse
          .Tag = "idx:" & i & ";state:" & False ' Use tag to store state information
          .Enabled = True
          End With
  Next
End


' Initialize a new game
Private Sub NewGame()
  Dim count, i, j As Integer
  Dim pzl, pval As String
  Dim pvals As String[]
  Dim lbl As Label
  Randomize
  ' Use Rand() for Gambas Release 3.6
  ' or CInt(Rnd()) for previous releases
  count = Rand(24)
  'count = CInt(Rnd(24.0))

  If bUseStoredGames Then
          'Randomly select puzzle
          count = Rand(NumberOfStoredGames)
          pzl = GetPuzzle(count)
          If pzl Then
          pvals = Split(pzl, ",")
          'Set the puzzle positions
          For Each pval In pvals
          i = Val(pval) - 1
          ChangeState(i)
          Next
          Else
          Message.Error("Puzzle Index Out of Range!")
          Endif
  Else
          ' Randomly set starting positions
          For i = 1 To count
          j = Rand(1, MaxStartingPositions)
          ChangeState(j)
          Next
  Endif
  Me.Refresh()
End


' Update the board after a move
Private Sub UpdateBoard()
  Dim x1, x2, y1, y2, i As Integer

  x1 = ActiveIndex - 1
  x2 = ActiveIndex + 1
  y1 = ActiveIndex - 5
  y2 = ActiveIndex + 5

  If x1 >= 0 And (x1 Mod 5) < 4 Then
          ChangeState(x1)
          aLbls[x1].Refresh()
  Endif
  If x2 < 25 And (x2 Mod 5) > 0 Then
          ChangeState(x2)
          aLbls[x2].Refresh()
  Endif

  If y1 >= 0 Then
          ChangeState(y1)
          aLbls[y1].Refresh()
   Else
          y1 = (y1 + 25) Mod 25
          ChangeState(y1)
          aLbls[y1].Refresh()
  Endif

  If y2 < 25 Then
          ChangeState(y2)
  aLbls[y2].Refresh()
  Endif

End


' Test for winning condition
Public Sub TestWin() As Boolean
  Dim i As Integer
  Dim flag As Boolean

  flag = True

  For i = 0 To aLbls.Length - 1
          If GetState(i) = True Then
          flag = False
          Endif
  Next

  Return Flag
End


' Process mouse click
Public Sub Labelgroup_MouseDown()
  Dim idx As Integer
  Dim state As Boolean
  Dim msg As String

  ActiveGame = True

  GetActiveLabel(Last)
  ChangeState(ActiveIndex)
  UpdateBoard()
  UpdateScore()
  If TestWin() Then
          msg = "You Won!!!\n" & "in " & tbTries.Text & " tries."
          Message.Info(msg)
  Endif
End


' Update Score text box
Private Sub UpdateScore()
  tbTries.Text = Str(Val(tbTries.Text) + 1)
End


'Get the label clicked on and save it
'in the active index and active state
Private Sub GetActiveLabel(lbl As Object)
  Dim data As String
  Dim datums, dats1, dats2 As String[]
  Dim idx As Integer
  Dim state As Boolean

  data = lbl.tag ' We stored state and index information in the tag...
  datums = Split(data, ";")
  dats1 = Split(datums[0], ":")
  dats2 = Split(datums[1], ":")
  ActiveIndex = dats1[1]
  ActiveState = dats2[1]
End


'Get the state of a label cell from the
'labels tag property
Private Sub GetState(idx As Integer) As Boolean
  Dim lbl As Label
  Dim data As String
  Dim datums, dats1 As String[]
  Dim state As Boolean

  lbl = aLbls[idx] ' Get the label

  data = lbl.tag ' We stored state and index information in the tag...
  datums = Split(data, ";")
  dats1 = Split(datums[1], ":")

  If CBool(dats1[1]) Then
	state = True
  Else
	state = False
  Endif
  Return state
End


'Set the state of the lable identified by the passed
'index value to state
Private Sub SetState(idx As Integer, state As Boolean)
  Dim lbl As Label

  lbl = aLbls[idx]
  If state Then
	lbl.tag = "idx:" & Str(idx) & ";state:" & True
	lbl.Background = iColorTrue
  Else
	lbl.tag = "idx:" & Str(idx) & ";state:" & False
	lbl.Background = iColorFalse
  Endif
End


'Toggle state of the label identified by idx
Private Sub ChangeState(idx As Integer)
  Dim state As Boolean

  state = GetState(idx)
  state = Not state
  SetState(idx, state)
End
```

```
'Process New Game button
Public Sub Button1_Click()
  ResetBoard()
  ActiveGame = False
  NewGame()
End


'Return a string of known good starting positions
'These are only 8 of the more than a 1000 starting
'positions with known solutions. Add more if you like
Private Sub GetPuzzle(n As Integer) As String
  Select Case n
            Case 1
            Return "1,3,4,5,6,8,10,12,13,14,16,18,19,21,23"
            Case 2
            Return "2,5,7,8,9,11,13,14,17,19,22,24"
            Case 3
            Return "1,2,3,5,6,7,8,12,13,15,16,17,19,20,21,22,23,24"
            Case 4
            Return "1,2,3,5,6,12,15,16,18,20,21,23"
            Case 5
            Return "4,6,9,15,16,17,18,23"
            Case 6
            Return "1,2,3,4,5,8,9,10,12,14,17,18,19,22,24"
            Case 7
            Return "3,6,9,10,11,13,18,19,24"
            Case 8
            Return "1,4,7,9,10,12,15,18,19,21,22,24"
            Case Else
            Return ""
  End Select

End


'Process check box for Use Stored Gamea
Public Sub cbUseStoredGames_Click()
  bUseStoredGames = cbUseStoredGames.Value
End
```

How to play

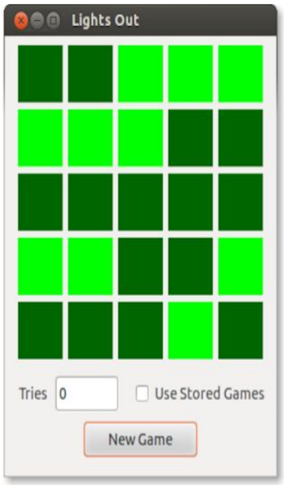
The Game of Lights is an interesting game to play. It is simple lightweight and runs with minimal support. It is developed using Gambas3. You can find the game in the github repository given at the end of the page.

The Game Interface consists of a 5*5 grid cells . At the start of the game, some of the cells will be lighted up and some others will be dark. The aim of the player is to turn the light off in every cell. Sounds easy ??? Well that is not all.

When you click on a cell its state is toggled. That is if the cell was already lit up it will turn darken ,else it wil lit up. Simultaneously four cells that are adjacent to cell on which you clicked will toggle its state ie, the cell to right, left, directly above and directly below the clicked cell will toggle their state. The player should try to turn the full 5*5 grid dark in minimum number of clicks.

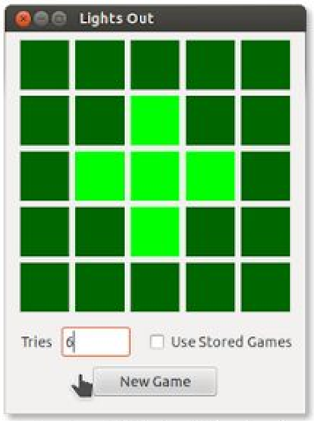
The game contains 8 pre stored initial condition setting. The player can get these settings by checking the box that says ‘use stored setting’.This will load any of the preset conditions to start the game. The algorithm may generate games that are not solvable. To avoid this problem we can choose any of the stored games. If you are really good at puzzles, then go for the random generated games. Enjoy playing.

Screenshots of the Game



Tries 0 □ Use Stored Games

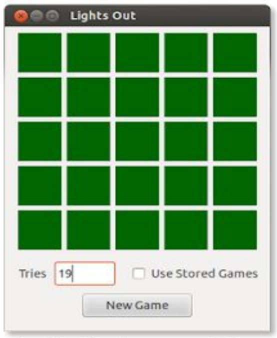
A random game generated



Tries 6 □ Use Stored Games

fig. 1 – Selected Cardinal Neighbors of Center Position.

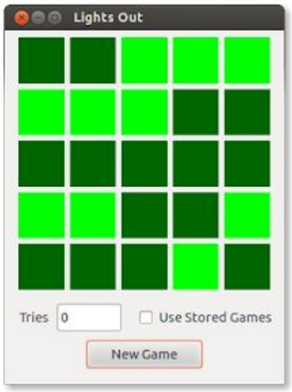
how the toggle when clicked



Tries 19 □ Use Stored Games

fig. 2 – To win you must turn out all positions on the board.

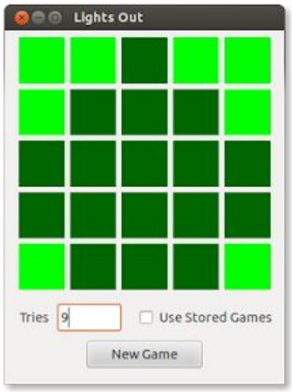
Win condition



Tries 0

□ Use Stored Games

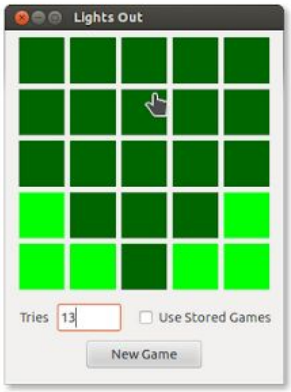
fig. 5 – Initial State Example



Tries 9

□ Use Stored Games

fig. 4 – Corner Case Top



Tries 13

□ Use Stored Games

fig. 3 – Corner Case Bottom

Examples of an initial state setting(I),Changes with top corners, and changes with bottom base corners.

Conclusion

We were able to successfully create a game called Lights Off using Gambas3.First of all,we learnt the basics of BASIC language and then about Gambas environment with the help of tutorial videos from youtube and documentations provided there . This project has helped us improve our knowledge of GUI programming.We sincerely thank our teachers to give us an opportunity to form groups and complete the project as we could share our ideas and have a fruitful discussions.

References

1.http://youtube.com/
2.http://stackoverflow.co.in/
3.http://www.gambasmag.com
4.http://gambaswiki.org/wiki/tutorial
5.http://gambas.sourceforge.net/en/main.html