CMSC 447

Software User Manual (SUM)

Name	Role	Signature
Daniel Sachs	Junior New Grad Associate Software Development Engineer I	
Innocent Kironji	Junior New Grad Associate Software Development Engineer I	
Sarah Alvarez	Junior New Grad Associate Software Development Engineer I	
Jason Schuler	Junior New Grad Associate Software Development Engineer I	
Pablo Burgos	Junior New Grad Associate Software Development Engineer I	
James Walls	Junior New Grad Associate Software Development Engineer I	

Chapter 1

Scope

1.1 Identification

“Game of Life: The Game”, henceforth referred to as “the game” is a software package developed and distributed by Group 2, also known as “As Long as We Graduate” and “Anything Goes” in Russell Cain’s CMSC 447 Spring ‘19 class. The game was developed to meet the specifications provided by Dr. Susan Mitchell, henceforth referred to as “the customer”.

1.2 Overview

The game is a desktop application. It is intended for use as a standalone tool on a single machine that allows two users to play against each other. The game runs a Game of Life simulation for each user with the object of “painting” a larger area than the opponent.

1.3 Overview

The purpose of this document is to provide a detailed description of the game's source code functionality.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 gol.Cell Class Reference

An individual cell on the game board.

Public Member Functions

- def __init__ (self, x, y, i, j, player)

  The constructor.
- def __str__ (self)

  Prints a readable message.
- def __repr__ (self)

  Prints an unambiguous message.
- def switchStatus (self)

  Switches the current status of the cell.

5.1.1 Detailed Description

An individual cell on the game board.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
def gol.Cell.__init__ (
              self,
              x,
              y,
              i,
              j,
              player )
```

The constructor.

Parameters

x	The x screen position of the cell.
y	The y screen position of the cell.
i	The row coordinate of the cell.
j	The column coordinate of the cell.
player	The owner of the cell.

5.1.3 Member Function Documentation

5.1.3.1 __repr__()

```
def gol.Cell.__repr__ (
              self )
```

Prints an unambiguous message.

Parameters

self	The current Cell object.

Returns

A string message describing the current state of the cell.

5.1.3.2 __str__()

```
def gol.Cell.__str__ (
              self )
```

Prints a readable message.

Parameters

self	The current Cell object.

Returns

A string message describing the current state of the cell.

5.1.3.3 switchStatus()

```
def gol.Cell.switchStatus (
              self )
```

Switches the current status of the cell.

A cell that is current alive will die, and a cell that is currently dead will become alive.

Parameters

self	The current Cell object.

The documentation for this class was generated from the following file:

- gol.py

5.2 gol.Game Class Reference

The main class of the game.

Public Member Functions

- def __init__ (self, canvas, root, pFrame, gameSpeed, player, global_turn, color='forest green', dead_↩ color='green2', cells_left=15, adversary=None)

  The constructor.
- def updateRemaining (self)

  Updates the number of remaining cells.
- def updateFrame (self)

  Function for updating the values for the player stats.
- def create_grid (self)

  This function creates the board on which the game will take place.
- def find_rect_coordinates (self, x, y)

  Finds the co-ordinates of the rectangle which has been clicked.
- def change_colour_on_click (self, event)

  Change the colour of the clicked grid and change the status of cell in the grid.
- def getBonus (self)

  Assigns bonus points to edit cells to the player in the lead.
- def paint_grid (self)

  Colors the cells in the grid.
- def changeInStatus (self, cell)

  Determines if the cell's status changes in the next gen.
- def begin (self)

  Begins the game.
- def stop (self)

  Stops the game.

5.2.1 Detailed Description

The main class of the game.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def gol.Game.__init__ (
                self,
                canvas,
                root,
                pFrame,
                gameSpeed,
                player,
                global_turn,
                color = 'forest green',
                dead_color = 'green2',
                cells_left = 15,
                adversary = None )
```

The constructor.

Parameters

canvas	The tkinter canvas.
root	The root.
pFrame	The statistics frame.
gameSpeed	The time between game ticks.
player	The current player.
global_turn	The Game turn.
color	The color of cells that are alive.
dead_color	The color of painted dead cells.
cells_left	The remaining cells the player can edit.
adversary	The other player.

5.2.3 Member Function Documentation

5.2.3.1 begin()

```
def gol.Game.begin (
                self )
```

Begins the game.

Parameters

self	The current Game object.

5.2.3.2 change_colour_on_click()

```
def gol.Game.change_colour_on_click (
              self,
              event )
```

Change the colour of the clicked grid and change the status of cell in the grid.

Parameters

self	The current Game object.
event	A particular tk event.

Returns

The updated Frame.

5.2.3.3 changeInStatus()

```
def gol.Game.changeInStatus (
              self,
              cell )
```

Determines if the cell's status changes in the next gen.

Parameters

self	The current Game object.
cell	The current cell.

Returns

Whether the cell changes in the next gen or not.

5.2.3.4 create_grid()

```
def gol.Game.create_grid (
              self )
```

This function creates the board on which the game will take place.

Parameters

self	The current Game object.

5.2.3.5 find_rect_coordinates()

```
def gol.Game.find_rect_coordinates (
              self,
              x,
              y )
```

Finds the co-ordinates of the rectangle which has been clicked.

Parameters

self	The current object.
x	The x coordinate of the clicked cell.
y	The y coordinate of the clicked cell.

5.2.3.6 getBonus()

```
def gol.Game.getBonus (
              self )
```

Assigns bonus points to edit cells to the player in the lead.

Parameters

self	The current Game object.

Returns

The number of bonus points awarded to the current player.

5.2.3.7 paint_grid()

```
def gol.Game.paint_grid (
              self )
```

Colors the cells in the grid.

Parameters

self	The current Game object.

5.2.3.8 stop()

```
def gol.Game.stop (
```

```
                self )
```

Stops the game.

Parameters

self	The current Game object.

5.2.3.9 updateFrame()

```
def gol.Game.updateFrame (
                self )
```

Function for updating the values for the player stats.

Parameters

self	The current Game object.

5.2.3.10 updateRemaining()

```
def gol.Game.updateRemaining (
                self )
```

Updates the number of remaining cells.

Parameters

self	The current Game object.

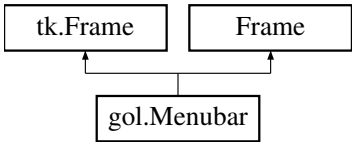
The documentation for this class was generated from the following file:

- gol.py

5.3 gol.Menubar Class Reference

Creates a menubar for the main display.

Inheritance diagram for gol.Menubar:



Public Member Functions

- def __init__ (self, parent)

  The constructor.
- def initUI (self)

  Initialized the user interface.
- def onView (self)

  Displays the information.
- def onExit (self)

  Closes the menu bar.
- def onGeneral (self)

  Displays general information.
- def onMenu (self)

  Displays Menu information.

5.3.1 Detailed Description

Creates a menubar for the main display.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 __init__()

```
def gol.Menubar.__init__ (
              self,
              parent )
```

The constructor.

Parameters

parent	The parent Frame.

5.3.3 Member Function Documentation

5.3.3.1 initUI()

```
def gol.Menubar.initUI (
              self )
```

Initialized the user interface.

Parameters

self	The current Menubar object.

5.3.3.2 onExit()

```
def gol.Menubar.onExit (
              self )
```

Closes the menu bar.

Parameters

self	The current Menu Bar object.

5.3.3.3 onGeneral()

```
def gol.Menubar.onGeneral (
              self )
```

Displays general information.

Parameters

self	The current Menu Bar object.

5.3.3.4 onMenu()

```
def gol.Menubar.onMenu (
              self )
```

Displays Menu information.

Parameters

self	The current Menu Bar object.

5.3.3.5 onView()

```
def gol.Menubar.onView (
              self )
```

Displays the information.

Parameters

self	The current Menu Bar object.

The documentation for this class was generated from the following file:

- gol.py

Chapter 6

File Documentation

6.1 gol.py File Reference

Contains class definitions for the game.

Classes

- class gol.Cell

  An individual cell on the game board.
- class gol.Menubar

  Creates a menubar for the main display.
- class gol.Game

  The main class of the game.

Functions

- def gol.SetSpeed (p1, p2, speed=200)

  Speeds up the game_speed based on the value found at the slider.
- def gol.end_of_game (p1, p2)

  Ends the game.

6.1.1 Detailed Description

Contains class definitions for the game.

Authors

Sarah Alvarez

Pablo Burgos

Innocent Kironji

Daniel Sachs

Jason Schuler

James Walls

6.1.2 Function Documentation

6.1.2.1 end_of_game()

```
def gol.end_of_game (
              p1,
              p2 )
```

Ends the game.

Parameters

p1	Player one's Game object.
p2	Player two's Game object.

6.1.2.2 SetSpeed()

```
def gol.SetSpeed (
              p1,
              p2,
              speed = 200 )
```

Speeds up the game_speed based on the value found at the slider.

Parameters

p1	Player one's Game object
p2	Player two's Game object
speed	The time between each game "tick" in nanoseconds.

6.2 main.py File Reference

The game driver.

Functions

- def main.promptWindow (prompt, info_list)

  Creates the window that will be displayed for the prompt and takes input.
- def main.getInfo (num_players)

  Getting color and name info for players.
- def main.displayWinner ()

  Displays the winner of the game.
- def main.begin_game (turnCounter, iters, turns, speed)

  Begins the game.
- def main.stop_game ()

  Stops the game.

- def main.set_speed (val)

  Sets the game speed.
- def main.updateDisplay (turnLabel, iters, turns, speed)

  Updates the display.
- def main.main ()

  The main driver function.

6.2.1 Detailed Description

The game driver.

Authors

Sarah Alvarez

Pablo Burgos

Innocent Kironji

Daniel Sachs

Jason Schuler

James Walls

6.2.2 Function Documentation

6.2.2.1 begin_game()

```
def main.begin_game (
              turnCounter,
              iters,
              turns,
              speed )
```

Begins the game.

Parameters

turnCounter	The turn counter for the game.
iters	
turns	The total number of turns for the game.
speed	The initial game speed.

6.2.2.2 getInfo()

```
def main.getInfo (
              num_players )
```

Getting color and name info for players.

Parameters

num_players	The number of players.

Returns

A list of info on the player' colors and names.

6.2.2.3 promptWindow()

```
def main.promptWindow (
              prompt,
              info_list )
```

Creates the window that will be displayed for the prompt and takes input.

Parameters

prompt	The prompt that will be displayed to the user.
info_list	The list to which the information will be appended.

6.2.2.4 set_speed()

```
def main.set_speed (
              val )
```

Sets the game speed.

Parameters

val	The time between game ticks in nanoseconds.

6.2.2.5 updateDisplay()

```
def main.updateDisplay (
              turnLabel,
              iters,
              turns,
              speed )
```

Updates the display.

Parameters

turnLabel	The turn counter for the game.

Parameters

iters	
turns	The total number of turns for the game.
speed	The initial game speed.

Index