JCEI's JAIHIND COLLEGE OF ENGINEERING, KURAN

DEPARTMENT OF AI & DS ENGINEERING

CERTIFICATE

This is to certify that the Manual under the subject
"Data Structure Laboratory (217531)"

SUBMITTED BY

Exam Seat No. 190842102

Is a Bonafede work carried out by student under the supervision of **Prof. Dumbre P. S.**
and it is

Submitted towards the partial fulfillment of the requirement of Second Year of AI & DS Engineering.

Prof. Dumbre.P.S	**Prof. Said S. K.**
Subject Teacher	**Head of Department**
Dept. of AI & DS Engineering.	**Dept. of AI & DS Engineering.**

Dr. D. J. Garkal
Principal

JCEI's JAIHIND COLLEGE OF ENGINEERING, KURAN

INDEX

Acknowledgement

We would take this opportunity to express my sincere thanks and gratitude to my teacher **Mrs** for his vital support and guidance in completing this project.

We also express our gratitude to all the facility members, parents and our fellow mates who have helped me in making this project a success. We also thank our almighty **God** for his blessings showed on me during this period.

Abstract

Snake and ladder is well known game among children even among matured people. The rules and regulation of the game are as well-known as the game. The case study meant for implementing this game without losing its interesting and attraction. The game is in two modes- two player and one player mode. In the one player the computer itself will act as the second player. The user can interact with the game using either keyboard or mouse. The number and position of the ladder and snake are generated fixed. Random mode numbering is used for user action in which the user is able to press the button long time and the number goes to a loop and when user released the number was displayed.

INTRODUCTION

The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product.

The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system.

In this project there are many details available related to the Snake & ladder game and its functions.

This is concerned with specifying equipment and software that will successfully satisfy the user requirement, in examining technical feasibility, configuration of the system is given more importance than the actual make of hardware.

Snake & ladder game is developed for DOS operating system and most of the latest version of Windows series OS is supported with DOS OS. Tools development for the use of the system is Turboc C compiler.

SYSTEM STUDY

Snakes and Ladders is an ancient Indian board game regarded today as a worldwide classic. It is played between two or more players on a gameboard having numbered, gridded squares.

A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start (bottom square) to the finish (Top Square), helped or hindered by ladders and snakes respectively.

The historic version had root in morality lessons, where a player's progression up the board represented a life journey complicated by virtues (ladders) and vices (snakes).

There was not a good GUI for Snake & ladder game for DOS operating system. We are trying to develop a system which makes, look and feel very interesting to play the game.

Snakes and Ladders originated in India as part of a family of dice board games, that included Gyan chauper and pachisi (present-day Ludo and Parcheesi). It was known as moksha patam or vaikunthapaali or paramapada sopaanam (the ladder to salvation). The game made its way to England and was sold as "Snakes and Ladders", then the basic concept was introduced in the United States as Chutes and Ladders (an "improved new version of England's famous indoor sport") by game pioneer Milton Bradley in 1943.

Proposed System

The proposed system was designed to give a professional look and feeling GUI for Snake & Ladder game. The system has a splash screen, which was animated and designed with good look and feel. Game board was drawn by taking all initial screen conditions and mouse interaction. Interrupts and sound functions are used for fast mouse interactions.

The program was divided in to two screens:

<u>Splash Screen</u>

- Program Splash and Details

<u>Game Window</u>

- Game square board.
- Dice display board.
- Dice button for running the dice.
- Single / Double Mode.

Game Play

Each player starts with a token on the starting square (usually the "1" grid square in the bottom left corner, or simply, the imaginary space beside the "1" grid square) and takes turns to roll a single die to move the token by the number of squares indicated by the die roll.

Tokens follow a fixed route marked on the game board which usually follows a boustrophedon (ox-plow) track from the bottom to the top of the playing area, passing once through every square. If, on completion of a move, a player's token lands on the lower-numbered end of a "ladder", the player moves the token up to the ladder's higher-numbered square. If the player lands on the higher-numbered square of a "snake" (or chute), the token must be moved down to the snake's lower-numbered square.

If a player rolls a 6, the player may, after moving, immediately take another turn; otherwise play passes to the next player in turn.

The player who is first to bring their token to the last square of the track is the winner.

Requirements

1 Software Requirements

Operating System	: DOS
Developing software	: IDE
Language used	: Pyhton
Library	: Graphics

2 Hardware Requirements

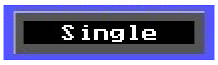
Processor	: PC with a Pentium II-class.
Memory	: 256 MB RAM.
Hard disk space	: 2 .5 GB installation drive.
Monitor	: 15 inch Color Monitor.
Keyboard	: 105 Keys.
Display Type	: Super VGA (800x600).
Mouse	: Serial Mouse.

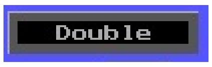
SYSTEM DESIGN

1 Input Design

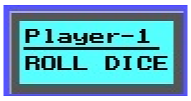
Input for the Snake & ladder program is dice running from each player. For player or user interaction the input design are designed with buttons and layout styles.

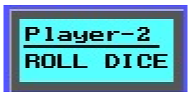
- **Game Mode**: Selecting player mode for the program there was two boxes allocated for players.



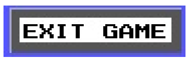


- **Dice Running:** For dice running two buttons are allocated for the players. Players need to hold the button and release when he needs to throw the dice. The received dice will be displayed on the on the dice box.





- **Exit / Stop Program**: The game can be stopped or exit with the click of Exit button.



- **New Game:** New game can be started with "New Game" button.

2 Output Design

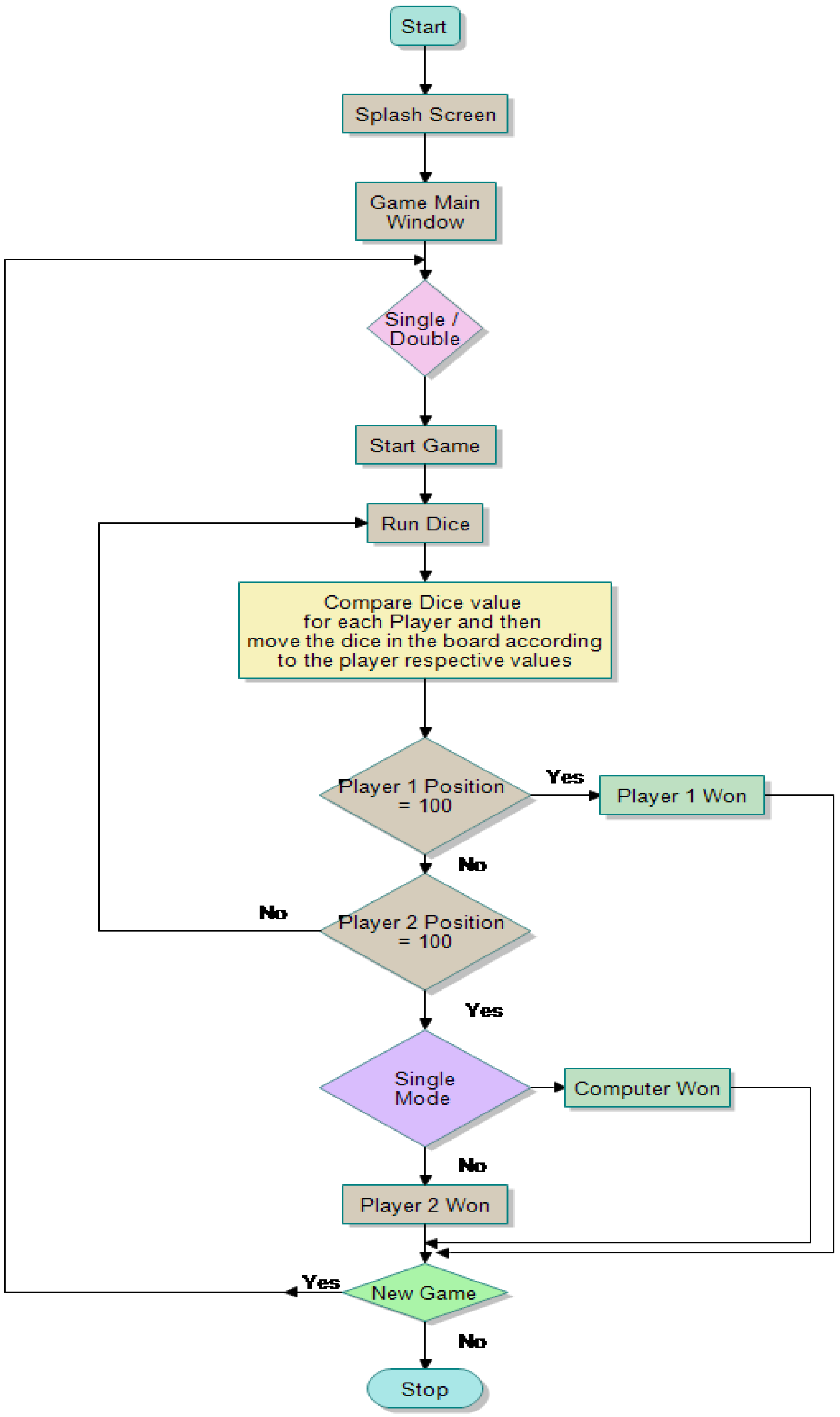
The output design was created with board design and 100 square cells. The Snake and ladder was also drawn on the bard game for moving and jumping dice icon of each player. Dice display board output the dice number received after each dice running action by each player. The total score is also displayed left side of the each dice running button.

3 Interface Design:

Interface design is the first step in the development phase for an engineering system. It is defined as "The process of applying various techniques and principles for the purpose of defining a process or a system in sufficient details to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used.

DOS operating system was not supported with an powerful GUI, so we tried to explore in giving a good GUI for our Snake & ladder program, which makes a good look and feel appearance for the players.

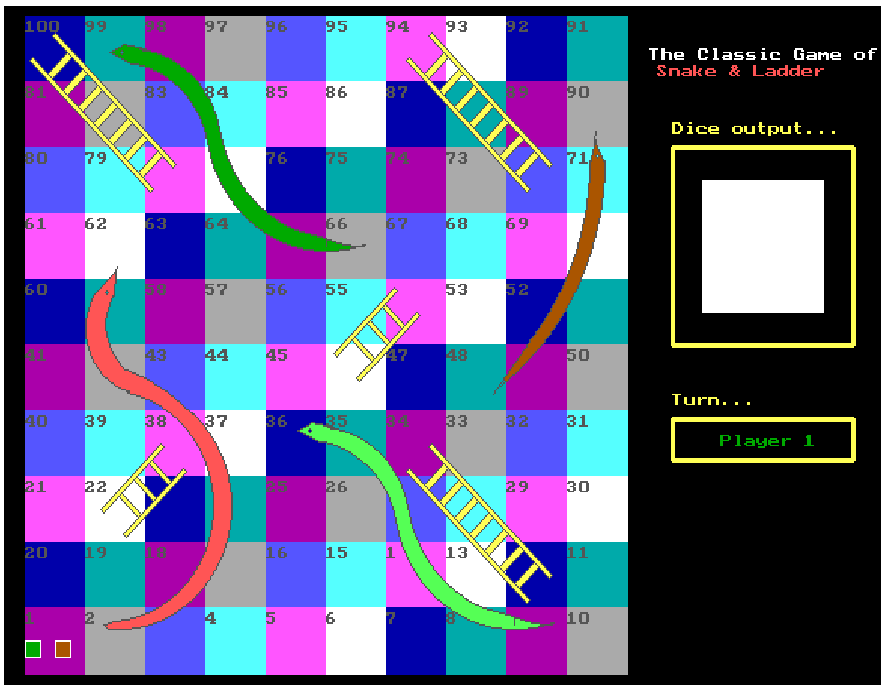
FLOWCHART


Start
Splash Screen
Game Main Window
Single / Double
Start Game
Run Dice
Compare Dice value for each Player and then move the dice in the board according to the player respective values
Player 1 Position = 100
Yes
Player 1 Won
No
No
Player 2 Position = 100
Yes
Single Mode
Computer Won
No
Player 2 Won
Yes
New Game
No
Stop

PYTHON PROGRAM

```
# GUI based snake and ladder game

#importing tkinter module

import random

from tkinter import *

import time

#function for generating random number on dice

def dice_roll():

    dice_number = random.randrange(1, 7, 1)

    return dice_number

#function for creating pawn for each player

def create_pawn(self, x, y, r, **kwargs):

    return self.create_oval(x-r, y-r, x+r, y+r, **kwargs)

Canvas.create_circle = create_pawn

#class for matcing the dice number with the snake and ladder

class matching_position():

    def find_snake_or_ladder(self, block, turn, position):

        x = 35*(turn>=3)

        y = (turn%3)*35

        if(block == 3):

           return 305+x, 150+y, 22

        elif(block == 5):

            return 545+x, 390+y, 8

        elif(block == 11):
            return 185+x, 30+y, 26
        elif(block == 20):
            return 545+x, 30+y, 29
        elif(block == 17):
           return 425+x, 510+y, 4
        elif(block == 19):
           return 665+x, 390+y, 7
        elif(block == 21):
           return 425+x, 390+y, 9
        elif(block == 27):
           return 65+x, 510+y, 1
        else:
            return position[0], position[1], block
#class for displaying the snake and ladder game
class game_board(object):
    def __init__(self,master,img):
        #Create board of snake and ladder
        board_width = 850
        board_height = 600
        self.color = ["#FFF", "#F00", "#0F0", "#00F", "#FF0", "#0FF"]
        self.canvas = Canvas(master, width = board_width, height = board_height, bg =
"brown")
        self.canvas.grid(padx=0, pady=0)
        self.canvas.create_image(360,300,anchor=CENTER, image = img)
        self.x = 65

        self.y = 510

        self.m = []

        self.num_player = "Players"

        self.player = []

        self.position = []

        self.i = 0

        self.block=[]

        self.dice_number = 1

        self.turn = 0

        #Menu for choosing number of players

        OPTIONS = ["Players", "2", "3", "4", "5", "6"]

        variable = StringVar(master)

        variable.set(OPTIONS[0]) # default value

        w = OptionMenu(self.canvas, variable, *OPTIONS, command=self.choose)

        w.pack()

        w.place(x=740, y=225)

        w.config(font=('calibri',(10)),bg='white',width=5)

        #Button for starting the game

        self.start_game = Button(self.canvas, text="Start", background='white', command
= self.start_game, font=("Helvetica"))

        self.start_game.place(x=770, y=400)

    #function for moving the player_pieces

    def start_game(self):

        if(self.num_player == "Players"):
            pass
        else:
            #dice_roll
            #Screen
            self.canvas.create_rectangle(810, 150, 760, 100, fill='white', outline='black')
            self.canvas.pack(fill=BOTH, expand=1)
            #Button
            self.diceRoll = Button(self.canvas, text="Roll",background='white',
                                   command = self.play_game, font=("Helvetica"))
            self.num_player = int(self.num_player)
            self.diceRoll.place(x=770, y=165)
            self.create_peice()
            self.start_game.place(x=-30, y=-30)
    #function for choosing number of players
    def choose(self, value):
        self.num_player = value
    #function for rolling the dice
    def rolling_dice(self, position, turn):
        dice_number = dice_roll()
        #dice_number = 1
        #Print dice_roll Value to screen
        dice_value = Label(self.canvas, text=str(dice_number),
                           background='white', font=("Helvetica", 25))
        dice_value.pack()
        dice_value.place(x=775, y=105)

        self.x, self.y = position[0], position[1]

        if(dice_number+self.block[turn] > 30):

            return [self.x, self.y

        self.dice_number = dice_number

        self.block[turn] += dice_number

              self.canvas.delete(self.player[turn])

        self.player_pieces(dice_number, turn)

        return [self.x, self.y]

     #function for moving the player_pieces in the board

    def player_pieces(self, dice_number, turn):

        #Starting value of and x and y should be 120 and 120

        #In create_circle initial value of x and y should be 100 and 550

        #To reach to the last block x should be 5*x and y should be 4*y

        #X should be added to value and Y should be subtracted

        # 5x120=600 and 4*120=480

        #m is the constant that tells which side to dice_number i.e. left to right or right to
left

        for i in range(dice_number,0,-1):

            self.x = self.x+120*self.m[turn]

            if(self.x>665 and turn < 3):

                self.y = self.y - 120

                self.x = 665

                self.m[turn] = -1

            elif(self.x>700 and turn >=3):
                self.y = self.y - 120
                self.x = 700
                self.m[turn] = -1
            if(self.x<65 and turn < 3):
                self.x = 65
                self.y -= 120
                self.m[turn] = 1
            elif(self.x<100 and turn >=3):
                self.x = 100
                self.y -= 120
                self.m[turn] = 1
            if(self.y<30):
                self.y=30
            # Code For the Animation of piece
            self.canvas.delete(self.player[turn])
            self.player[turn] = self.canvas.create_circle(self.x, self.y, 15, fill=self.color[turn],
outline=self.color[turn])
            self.canvas.update()
            time.sleep(0.25)
        print(self.x, self.y, self.block[turn])
        self.x, self.y, self.block[turn] =
matching_position().find_snake_or_ladder(self.block[turn], turn, [self.x, self.y])
        if(any(self.y == ai for ai in [390, 425, 460, 150, 185, 220])):
            self.m[turn] = -1
        else:
            self.m[turn] = 1
        print(self.x,self.y, self.block[turn])
        self.canvas.delete(self.player[turn])
        self.player[turn] = self.canvas.create_circle(self.x, self.y, 15, fill=self.color[turn],
outline="")
    #function for creating the player_pieces
    def create_peice(self):
        for i in range(int(self.num_player)):
            if(i==3):
                self.x += 35
                self.y -= 105
            self.player.append(self.canvas.create_circle(self.x, self.y, 15, fill=self.color[i],
outline=""))
            self.position.append([self.x, self.y])
            self.m.append(1)
            self.block.append(1)
            self.y += 35
    #function for playing the game
    def play_game(self):
        if(self.dice_number == 6):
            turn = self.turn
        else:
            turn = self.i%self.num_player
            self.i += 1
            self.turn = turn
        self.position[turn] = self.rolling_dice(self.position[turn], turn)

        if(self.block[self.turn] >= 30):

            self.diceRoll.place(x=-30, y=-30)

            print("Won", self.turn+1)

            top = Toplevel()

            top.title("Snake and Ladder")

            message = "Player " + str(self.turn+1) + " Won"

            msg = Message(top, text=message)

            top.geometry("%dx%d%+d%+d" % (100, 100, 250, 125))

            msg.pack()

            button = Button(top, text="Dismiss", command=top.destroy)

            button.pack()

#defining the main function

def main():

    master = Tk()

    master.title("Snake and Ladder")

    master.geometry("850x600")

    img = PhotoImage( file = "snake-ladder-board.gif")

    x = game_board(master,img)

    master.mainloop()

main(
```

OUTPUT



CONCLUSIONS

Throughout this thesis our aim was to develop a Snake & Ladder game that allows user to interact with the game using mouse.

The entire game is explained and displayed in a well arranged GUI. Two players are given equal chance for winning the game. Single mode and Double mode allows using computer as second player.

REFERENCES

- www.pythonprogramming.com
- www.cs.cf.ac.uk/Dave/C/CE.html
- http://www.brackeen.com/vga/index.html
- http://www.brackeen.com/vga/index.html
- www.programmingsimplified.com/p-graphics-programming- tutorial.html