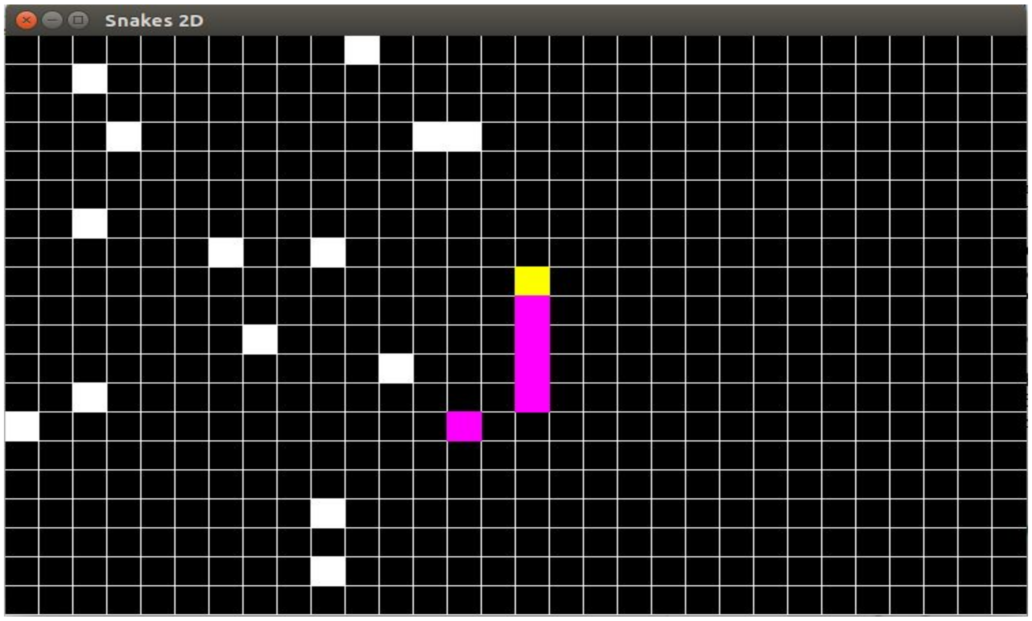
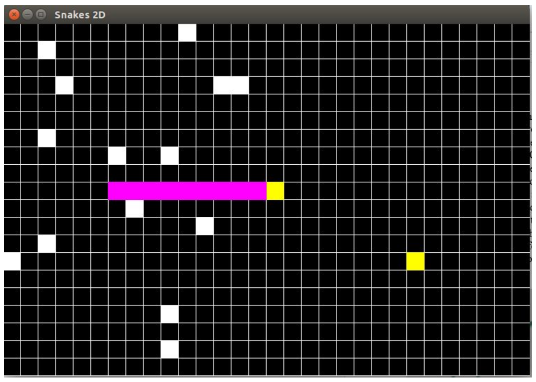
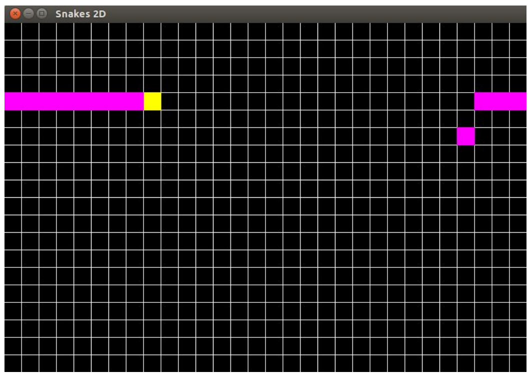
Snake

Report for the Graphics and Visual Computing project.







Project By:
Lakshay Gupta (IIT2014044)
Akshay Malhotra (IIT2014018)
Tushar Sudhakar Pedapalliwar (IIT2014059)

CANDIDATE'S DECLARATION

We certify that the work embodied in this project is our own bonafide work carried out by us under the supervision of Dr. Pavan Chakraborty and Dr. Satish Singh during the fifth semester at the Indian Institute of Information Technology, Allahabad. The matter embodied in this work has not been submitted for the award of any other degree/ diploma. We declare that we have faithfully acknowledged, given credit to and referred to the research workers wherever their works have been cited in the text and the body of the project. We further certify that we have not wilfully lifted up some other's work, paragraph, text, data, results, etc. reported in the journals, books, magazines, reports, dissertations, thesis etc. or material available at web-sites and included them in this project and cited as our own work.

Lakshay Gupta (IIT2014044)
Akshay Malhotra (IIT2014018)
Tushar Sudhakar Pedapalliwar (IIT2014059)
Dated: 17th of November, 2016.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Pavan Chakraborty and Dr. Satish Singh at the Indian Institute of Information Technology, Allahabad for sharing their pearls of wisdom with us during the course of this research.

We are also immensely grateful to our friends for their comments on an earlier version of the manuscript and providing insight and expertise that greatly assisted the project.

We are also thankful to the institute for providing us with the necessary resources required for the timely completion of the project.

INTRODUCTION

Snake is a classic 90's game that has been developed on various platforms. It rose to fame with the skyrocketing reputation of Nokia in early 2000's as it was included was a built-in game with their mobile devices. Even till date, advanced android based applications of this classical piece are developed and released on the Play Store with hopes of younger generation getting a taste of the old times. No matter on which part of the globe you're at, everyone's heard of snake and has played it.

A basic version of this classic game is presented by our team as part of the course requirement for the partial fulfillment of Graphics' and Visual Computing.

PROBLEM DEFINITION

The game rotates on the basic idea that any living organism required food and security as a basic right for living. In this classical game, a snake travels through different regions of the grid in search for food. However, the grid is randomly scattered with wall block that prevent the snake to go through it. Therefore, the snake has to dodge these walls and make it's way to the food block. If the snake happens to collide with a wall block, the game ends. However, if the snake successfully acquired a block of food, it grows in length and has to procure more food for its existence.

In this game, the grid is thought to be circular. This means that once the snake travels to the extreme right end of the grid, it appears from the left side of the grid.

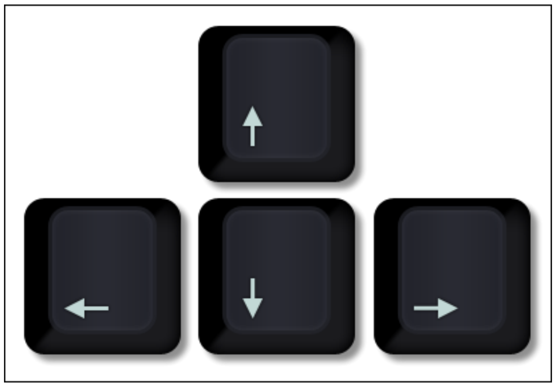
FEATURES

1. Multiple colors for bricks, food and the snake.
2. Wall blocks are randomly distributed throughout the grid.
3. Length of snake increase with procurement of food block, thereby, increasing the difficulty level as it gets harder for the snake to travel.
4. The grid is assumed to be circular.
5. Snake cannot collide with itself or a wall block.

INSTRUCTIONS

Snake is an interactive and dynamic game in which the player is represented by a snake and should procure as much of the food blocks possible without smashing itself into a wall block or itself. The player's score rises with the gaining more food blocks.

- After the initial screen has been loaded, the game is to be started automatically with the snake moving towards a safe direction, away from the wall blocks.
- The movements of the snake is controlled with the help of the 4 direction buttons on the keyboard, ie,



Up Arrow - To move the snake upwards

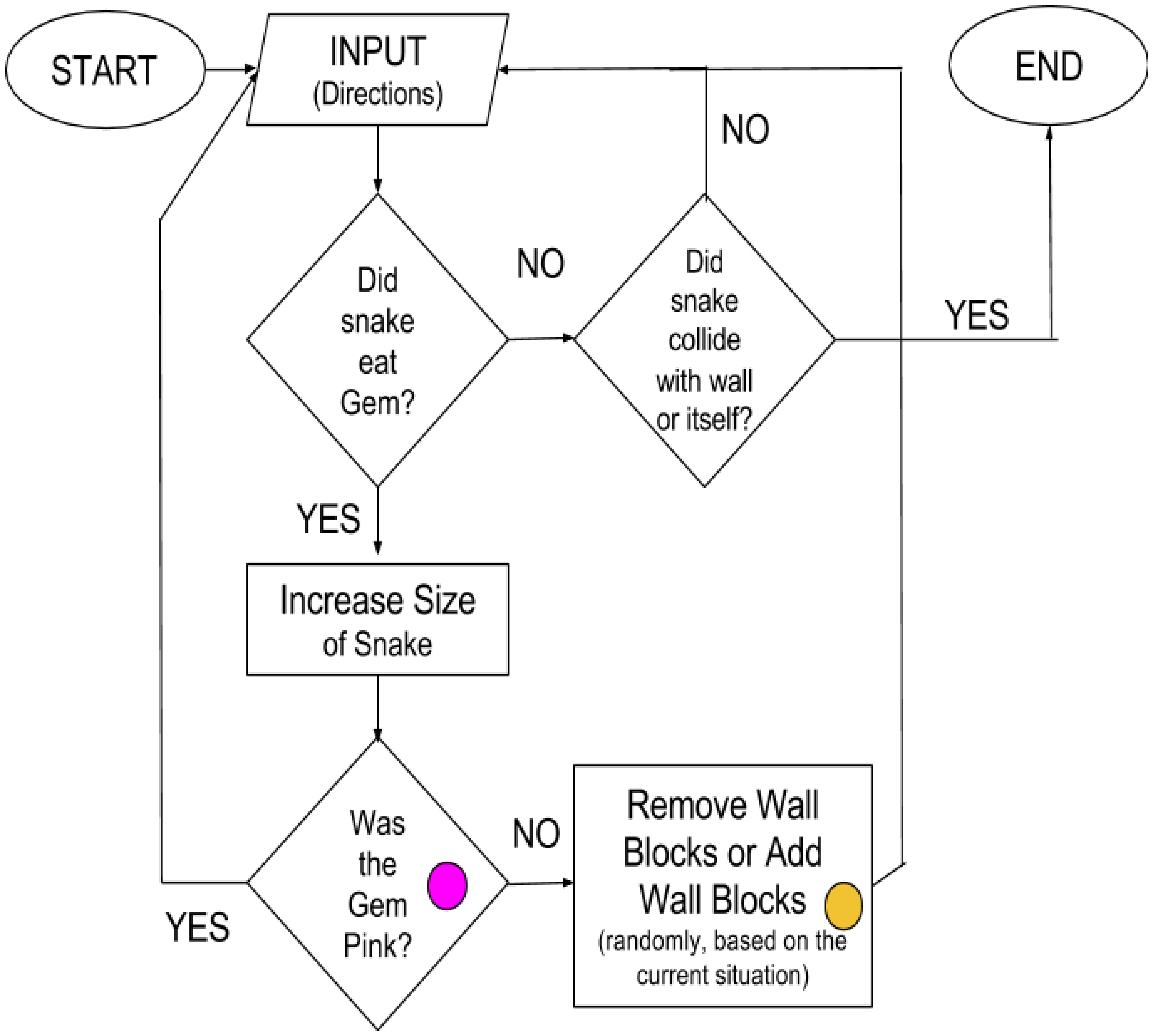
Down Arrow - To move the snake downwards

Right Arrow - To move the snake right

Left Arrow - To move the snake left.

- The player's aim is to collect the most amount of food blocks without colliding with itself or the wall blocks.

Flowchart / Concept of Working Model


START
INPUT
(Directions)
END
NO
Did snake eat Gem?
NO
Did snake collide with wall or itself?
YES
YES
Increase Size of Snake
Was the Gem Pink?
NO
Remove Wall Blocks or Add Wall Blocks
(randomly, based on the current situation)
YES

Flowchart of Snake

PROGRAM

game.hpp

```
int obsx[20];//array storing x co-ordinates of
obstalces
int obsy[20];//array storing y co-ordinates of
obstacles
int count=0;//count checks how many gems were
collected
int obsflag=1;//flag to keep or remove obstacles
int special;//checks if the gem is special or not
class fruct{
   public:
        int x, y;//x and y co-ordinate of gem
        void gen() {//generator function for gems
        if(sh%5==0 && sh!=5)
        {
                special=1;
        }
        x = rand() % VM_N;
        y = rand() % VM_M;

        for(int i = 0; i < sh; i++)
                if(s[i].x == x && s[i].y == y)
                        gen();
        for(int i=0;i<20;i++)
        {
        if(obsx[i]==x && obsy[i]==y)//make sure
gem is not positioned on an obstacle
                gen();
        }
   }


   void draw() {//drawing funtion of gems
        if(special==1)
                glColor3f(1,1,0);
        else
                (1, 0, 0);
        glRecti(x * VM_Scale - 1, y * VM_Scale, (x +
1) * VM_Scale, (y + 1) * VM_Scale + 1);
   }
};

fruct fru;

class game {
   public:
        void over() {
                std::ofstream RO("save.snk");
                RO << myMX;
                RO.close();
                exit(0);
        }
        void drawField() {
                glColor3f(1,1,1);
                glBegin(GL_LINES);//drawing a
grid using white lines

 for(int i = 0; i < VM_N; i++) {
                        glVertex2i(i * VM_Scale,
0);    glVertex2i(i * VM_Scale, VD_H);
                }
                for(int i = 0; i < VM_M; i++) {
                        glVertex2i(0, i *
VM_Scale);    glVertex2i(VD_W, i * VM_Scale);
                }
                glEnd();
                for(int i=0;i<20 && obsflag==1
;i++)//drawing obstacles
                {
                        glColor3f(1,1,1);
                        glRecti(obsx[i] *
VM_Scale - 1, obsy[i] * VM_Scale, (obsx[i] + 1) *
VM_Scale, (obsy[i] + 1) * VM_Scale + 1);
                }
                                glEnd();
        }

        void drawSnake() {//drawing snake
                glColor3f(1, 1, 0);
                glRecti(s[0].x * VM_Scale - 1, s[0].y
* VM_Scale, (s[0].x + 1) * VM_Scale, (s[0].y + 1) *
VM_Scale + 1);
                for(int i = 1; i < sh; i++)
                        {glColor3f(1, 0, 1);
                        glRecti(s[i].x * VM_Scale -
1, s[i].y * VM_Scale, (s[i].x + 1) * VM_Scale, (s[i].y +
1) * VM_Scale + 1);}
        }

        void stroke() {//refreshing the grid
                if(count==1 && special==1)
                {
                        special=0;
                        fru.gen();
                        count=0;
                }
                for(int i = sh; i > 0; i--) {
                        s[i].x = s[i-1].x;
                        s[i].y = s[i-1].y;
                }

                if(dir == 1)
                        s[0].y += 1;
                if(dir == 2)
                        s[0].x += 1;
                if(dir == 3)
                        s[0].y -= 1;
			if(dir == 4)
				s[0].x -= 1;

			if(fru.x == s[0].x && fru.y == s[0].y
) {//checking if gem is eaten
				sh++;
				if (special == 1 &&
obsflag==1)
				{
					obsflag=0;
					special=0;
				}
				if(special==1 &&
obsflag==0)
				{
					obsflag=1;
					assign();
					special=0;
				}
				fru.gen();
				if(sh > MX)
					myMX = sh;

				if(step > 1)
					step -=5;
			}

			for(int i=0;i<sh;i++)//circular
movement
			{

s[i].x=(s[i].x+VM_N)%VM_N;

s[i].y=(s[i].y+VM_M)%VM_M;
			}

			for(int i=1;i<sh;i++)//if snake
crashes with itself
			{
				if(s[0].x==s[i].x &&
s[0].y==s[i].y)
					over();
			}
			for(int i=0;i<20&& obsflag==1;i++)
			{
				if(obsx[i]==s[0].x &&
obsy[i]==s[0].y)//if snake crashes with an obstacle
					over();
			}


		}

		void assign()//assign random position to
obstacles
		{
			for(int i=0;i<20;i++)
			{
				obsx[i]=rand()%VM_M;
				obsy[i]=rand()%VM_N;
			}
		}

		static void keyboard(int key, int a, int b)
{//keyboard input
			switch(key) {
				case 101:
					if(dir != 3)
						dir = 1;
				break;

				case 102:
					if(dir != 4)
						dir = 2;
				break;

				case 103:
					if(dir != 1)
						dir = 3;
				break;

				case 100:
					if(dir != 2)
						dir = 4;
				break;
			}
		}
}
```

vars.hpp

```
#define VM_N 30
#define VM_M 20
#define VM_Scale 25

#define VD_W VM_N * VM_Scale
#define VD_H VM_M * VM_Scale

struct snk {
   int x;
   int y;
};
int dir = 1, sh = 5;
snk s[100];
int step = 150;
int MX, myMX;
```

main.cpp

```
#include <GL/glut.h>
#include <stdlib.h>
#include <fstream>
#include "vars.hpp"
#include "game.hpp"

game snake;

void display() {
   glClear(GL_COLOR_BUFFER_BIT);
   snake.drawField();
   snake.drawSnake();
   fru.draw();
   glutSwapBuffers();
}

void timer(int = 0) {
   display();
   snake.stroke();
   glutTimerFunc(step, timer, 0);
}

void firstRecordSetup() {
   std::ifstream RI("save.snk");
   RI >> MX;
   myMX = MX;
}

void firstGameSetup() {
   for(int i = 0; i < sh; i++) {
           s[i].x = VM_N / 2;
           s[i].y = (VM_M + sh) / 2 - i;
   }

   fru.gen();
}



void firstSetup() {
   glClearColor(0, 0, 0, 0);
   glColor3f(1, 1, 1);
}

int main(int argc, char **argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE |
GLUT_RGBA);
   glutInitWindowSize(VD_W, VD_H);
   glutInitWindowPosition(100, 100);
   glutCreateWindow("Snakes 2D");
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, VD_W, 0, VD_H);


   firstSetup();
   firstGameSetup();
   firstRecordSetup();
   snake.assign();

   glutSpecialFunc(game::keyboard);
   glutDisplayFunc(display);
   glutTimerFunc(step, timer, 0);
   glutMainLoop();

   return 0;
}
```

TOOLS AND LANGUAGES USED

The following tools were used to give the project its final shape and form-

1. Linux operating system for coding and testing
2. OpenGL library
3. C++ programming language environment
4. Sublime text 3 code editor
5. Git and Github VCS for individual contributions to the shared project

---

RESULTS AND CONCLUSIONS

Successfully implemented the game with several additional features such as the existence of a circular grid and wall block surprisingly appearing/disappearing apart from the basic functionalities in the classical game. The game may remind you of your childhood.

REFERENCES

1. Computer Graphics- Hearn and Baker
2. OpenGL documentation- https://www.opengl.org/wiki/Getting_Started
3. http://www.cprogramming.com/tutorial/opengl_introduction.html
4. https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxza3NpaWl0YXxneDoyMjk3Nzc2Yjk4YWRhYjUz

---