

**Hungry Cat**

**An Opengl 2-D Game**

Summer 2019-2020

Department of Computer Science

American International University-Bangladesh

Sec: L

Dipta Justin Gomes

Computer Graphics

**GROUP MEMBERS**

1. Rahman, Shojibur 18-36205-1

2. Brishti, Anoara Akter 18-36116-1

3. Akand, Riyad Hossain 18-36736-1

4. Omi, Kazi Shahriar Islam 17-35824-3

5. Masum, Medul Hossain 17-35792-3

**Submission: Monday, September 21, 2020**

**Introduction:**

The goal of our game project is to design a 2-D graphical computer game using OpenGL. For our project, we decided to design a 2-D maze game where the objective of the game is to find a way to escape out of the maze. The user, played as a cat, has to find a way to get out of the dogs’ dungeon without being caught. The game is designed in a Windows environment and written in C/C++ OpenGL. As a result, we have created a 2-D maze game that is fun and enjoyable.

**Background Study:**

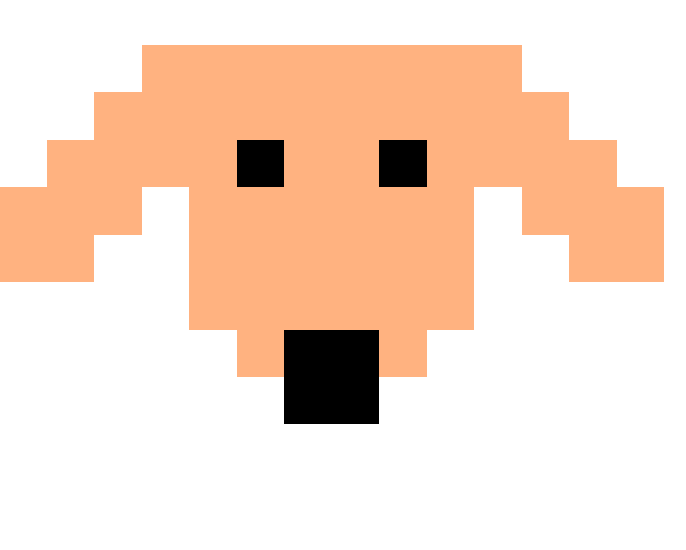
The primary objective of the game is for the user to search for a way out of a 2-D dogs’ dungeon. Throughout the maze, there are four dogs that the user must avoid in order to exit the maze successfully.

In the game, the player is played as a cat that has to find its exit through a dog dungeon. To escape from the maze, the player must avoid getting caught by dogs in different areas in the maze.



The major character of the game that represents the player. The Cat would be visible when player is playing the game and *The Hungry Cat* uses a third person view.

The villain of the game. There are multiple walking dogs in the maze. Player should avoid the dogs if they see them.



Maze exit (Kitchen door). Player complete game level by going through the exit

**User Manual:**

The player controls the game play using particular keys on the keyboard. The different movements of the cat available for the player are listed below.

Direction Keys:

Key ‘w’: Move the cat forward

Key ‘s’: Move the cat backward

Key ‘a’: Turn left

Key ‘d’: Turn right

Other:

Key ‘END’: Terminate the game

Key ‘2’: Terminate the game

Key ‘HOME: Restart the game

Key ‘0’: Start Story mode

Key ‘1’: Start New game

**Game Design:**

The cat, dogs and other items in the game are drawn with different shapes and lines using various GLUT programming functions. These basic shapes are then altered into a desired form using transformation functions including glTranslate and glScalef. For example, the following code shows how the cat’s body can be drawn with basic shapes and transformation.

glPushMatrix ();

glTranslatef (2, 2, 0.0f);

glScalef (.2,.2,0.0f);

glBegin (GL\_POLYGON);

glColor3f (.9, 0.1, .5);

glVertex2f (7, 0);

glVertex2f (7, 1);

glVertex2f (8, 1);

glVertex2f (8, 0);

glEnd ();

glPopMatrix ();

**Collision Detection:**

Since there are various objects located in the maze, the implementation of collision detection is required to detect when the player hits the wall of the maze or other items in the maze. There are three types of collision detection that we are concerned with in the game: collision between the dog and cat, collision between the dog and wall and collision between the cat and wall. For the first type, dog is bounded by a rectangular area. A collision is detected when the edges of the two objects’ intercept one another one (dog and cat). The second type of collision, between the dog and a wall, is slightly different due to the fact that the maze is built up by numerous walls. To simplify this check, we isolate the dog to a limited area. There are four dogs so only collisions these four walls will need to be checks. A collision occurs when the cat’s area extends across the border of the area and the wall on the corresponding side of the area has not been removed. The condition for a collision is: for two dimension (i.e. x, y).

**Sound Effects**

The sound clip are in wav format which are loaded and stored in a same location for easy access. The sound are used as background music for the game.

**Implementation:**

**Functions we use:**

startscreen()

winscreen()

gameover()

output(parameters)

story()

idle()

updateDog()

updateDog2()

updateDog3()

updateDog4()

dog()

dog\_bone()

dog\_home()

cat()

end\_pont()

display()

handleKeypress(parameters)

SpecialKey()

main(parameters)

**Logical Codes:**

void updateDog(int value) {

if((posi > 24) && (flag == 0)){

sped = .4f;

posi -= sped;

flag = 0;

}

else if((posi <= 24) && (posi >= 23)){

sped = .02f;

posi -= sped;

flag = 1;

}

else if((posi < 23) && (flag == 1)) {

posi = 24.1;

sped = .4f;

posi += sped;

flag = 2;

}

else if((posi > 24) && (posi < 46) && (flag == 2)) {

sped = .4f;

posi += sped;

if(posi > 45){

flag = 3;

} else {

flag = 2;

}

}

else if ((posi > 45) && (flag == 3)){

posi = 45;

sped = .4f;

posi -= sped;

flag = 0;

}

glutPostRedisplay();

glutTimerFunc(100, updateDog, 0);

}

void idle(){

if (state == 1){if (((position >= 51 && position <= 56) && (positionup >= 53 && positionup <= 58))){state = 4;}}

if(state == 1){if ((position < 6.8 && position >= posD2) && (positionup <= (posupD2+2) && positionup >= (posupD2-2))){state = 5;}}

if(state == 1){if ((position < 28.5 && position >= posD3) && (positionup <= (posupD3+2) && positionup >= (posupD3-2))){state = 5;}}

if(state == 1){if ((position > 49.9 && position <= posD4+3) && (positionup <= (posupD4+2) && positionup >= (posupD4-2))){state = 5;}}

if(state == 1){if ((positionup < 5.3 && positionup >= posiu) && (position <= (posi+4) && position >= (posi-2))){state = 5;}}

glutPostRedisplay();

}

void handleKeypress(unsigned char key, int x, int y) {

switch (key) {

case 'a':

if(position < 2){

moveflag = 1;

}

else if(positionup > 3.6 && positionup < 9.2){

x = position - 0.8;

if(position > 3.6 && position < 17.2){

moveflag = 1;

}

else{

position -= .8;

cout<<position<<',';

cout<<positionup<<endl;

}

}

else {

position -= .8;

cout<<position<<',';

cout<<positionup<<endl;

}

break;

case 'd':

if(position > 54){

moveflag = 1;

}

else if(positionup > 3.6 && positionup < 9.2){

x = position + .8;

if(x > 3.6 && x < 17.2){

cout<<position<<endl;

}

else {

position += .8;

cout<<position<<',';

cout<<positionup<<endl;

}

}

else {

position += .8;

cout<<position<<',';

cout<<positionup<<endl;

}

break;

case 'w':

if(positionup > 106){

moveflag = 1;

}

else if(position > 3.6 && position < 16.4){

y = positionup + .8;

if(y > 3.6 && y < 10){

cout<<positionup<<endl;

}

else {

positionup += .8;

moveflag = 0;

cout<<position<<',';

cout<<positionup<<endl;

}

}

else {

positionup += .8;

moveflag = 0;

cout<<position<<',';

cout<<positionup<<endl;

}

break;

case 's':

if(positionup < 2){

moveflag = 1;

}

else if(position > 3.6 && position < 16.4){

y = positionup - .8;

if(y > 3.6 && y < 9.2){

moveflag = 1;

} else {

positionup -= .8;

cout<<position<<',';

cout<<positionup<<endl;

}

} else {

positionup -= .8;

cout<<position<<',';

cout<<positionup<<endl;

}

break;

case '1':

state = 1;

position = 2;

positionup = 2;

break;

case '2':

exit(1);

break;

case '0':

state = 6;

break;

case ' ':

state = 1;

position = 2;

positionup = 2;

break;

glutPostRedisplay();

}

}

**Conclusion:**

In the game The Hungry Cat, we have implemented a game environment that includes 2D viewing and objects. To enhance the game environment we have also implemented advanced features such as collision detection and sound effects using different OpenGL libraries available. The variety of game features such as dogs, maze difficulties and background music makes the game more challenging, interesting and enjoyable to play.