R2D2: A BGI GRAPHICS SINGLE PLAYER GAME

SUBMITTED BY

Rishab Arora

06111503109

IT 4th Semester

Introduction

About the game

R2D2 is a third person single player shooter. It has been implemented using BGI for DOS bundled with the Borland Turbo C++ v3.0 compiler. There are two player: Human and Computer. Both players can fire Energy Orbs at each other in the direction they are facing. The objective is to kill the other player.

Players

1) Han Solo [Human Player]

Han Solo is the protagonist and the Human Player. It can be controlled by the following hard coded controls:

‘W’ = UP

‘A’ = LEFT

‘S’ = DOWN

‘D’ = RIGHT

SPACE = FIRE

‘U’ = QUIT THE GAME

2) Darth Vader [Computer Player]

Darth Vader is the computer player. No AI per se has been programmed and difficulty is due to fast switching of the randomized movements of the character

A check on Darth’s location mitigates his chances of leaving the arena.

Darth Vader moves in a direction for 10 units of gameplay, fires a bullet (Energy Orb) and changes direction to a random value. This quick switching makes it’s movements highly erratic and unpredictable.

Energy Orbs

Also called Bullets. These are spheres that emerge from the center of the player and attain the direction of the player. They travel at a constant speed until they reach either the boundary or hit a player.

Drawing Technique

Character (Player/Computer)

The character is drawn by a combination of ellipses, rectangles and pie-slices. Both players have their outline and fill coded in their classes which are a derivation of the ‘character’ class which includes elementary data about the characters.

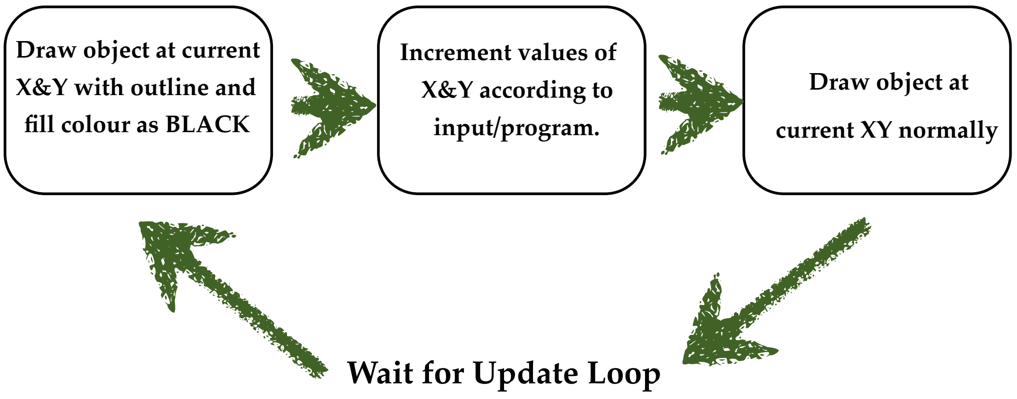
Energy Orbs

The energy orbs (class name Bullet) are of two types, one for player and one for enemy. Each bullet calls for collision detection and modifies the health of the opposite player in case of a hit.

FLICKERING

The Borland Graphics Interface in Turbo C++ v3.0 was not designed to support a high frame rate. Hence, if the cleardeive() or clearviewport() command is used to often, the whole view starts to flicker. To prevent flickering of constant or static elements of the arena, a very unique drawing system has been implemented.

However, flickering of active objects seems to remain constant.


Draw object at current X&Y with outline and fill colour as BLACK
Increment values of X&Y according to input/program.
Draw object at current XY normally
Wait for Update Loop

Game Loop

Every game has a fixed or variable step update loop, which is essentially an infinite loop. This particular game has a fixed length duration loop of 0.01 seconds.

Infinite Loop Steps:

1. If time difference > 0.01 seconds
    a. update bullet position
    b. update move_count. If move_count>10 ,
        i. Fire Enemy Bullet
        ii. Randomize Enemy Direction
        iii. Reset move_count to 0
2. If any key is hit, intercept it and evaluate
    a. ‘u’ to quit
    b. ‘SPACE’ to shoot
    c. else move Player in chosen direction
3. update clock

SOURCE CODE

Memory Usage: approx. 2 MB

Lines of Code: 374

```
#include <graphics.h>    // graphics library declarations
#include <iostream.h>
#include <conio.h>       // for getch() function
#include <stdlib.h>
#define LEFT -1
#define UP 0
#define DOWN 1
#define RIGHT 2
#define MAGNFY 5
#include <dos.h>
#include <time.h>

int bullet_count=0;
int bulletE_count=0;

class character{
        public:
        int x, y,dir;
        int width, height;
};

int ex,ey,px,py;
int ehealth, phealth;

int collisionEnemy(int,int);
int collisionPlayer(int,int);

class bullet{
        int x,y,dir;
        int active;
        public:
        void deploy(int a, int b, int dirin){
        x=a; y=b; dir = dirin;
        active=1;
        setcolor(BLACK);
           setfillstyle(1,BLACK);
           fillellipse(x,y,4,4);
        }
        void hit(){
        setcolor(BLACK);
           setfillstyle(1,BLACK);
           fillellipse(x,y,4,4);
         active=0;
        }
        void draw(){
        if (active==0) return;
        setcolor(BLACK);
           setfillstyle(1,BLACK);
           fillellipse(x,y,4,4);
```

```
            if (dir==LEFT) x-=2;
            if (dir==RIGHT) x+=2;
            if (dir==UP) y-=2;
            if (dir==DOWN) y+=2;
            setfillstyle(1,WHITE);
            fillellipse(x,y,4,4);
            //delay(10);
            if (collisionEnemy(x,y)==1) hit();
            if ((x>getmaxx())||(x<0)||(y>getmaxy())||(y<0)) active=0;
        }

} bullet[100];


class bulletE{

        int x,y,dir;
        int active;
        public:
        void deploy(int a, int b, int dirin){
        x=a; y=b; dir = dirin;
        active=1;
        }
        void hit(){
         active=0;
        }
        void draw(){
           if (active==0) return;
           setcolor(BLACK);

            setfillstyle(1,BLACK);
            fillellipse(x,y,4,4);

            if (dir==LEFT) x-=2;
            if (dir==RIGHT) x+=2;
            if (dir==UP) y-=2;
            if (dir==DOWN) y+=2;
            setfillstyle(1,BLUE);
            fillellipse(x,y,4,4);
            //delay(10);
            if (collisionPlayer(x,y)==1) hit();
            if ((x>getmaxx())||(x<0)||(y>getmaxy())||(y<0)) active=0;
        }

} bulletE[200];

class enemy: public character{
int dely;
 public:
        enemy(int a, int b){
                dely=0;
                dir = RIGHT;
                x=a;
                y=b;
                ehealth=20;
        }
```

```
void draw(){
//cout<<x<<", "<<y<<endl;

        int body[8];
        setcolor(BLACK);

        body[0]=x-15;
        body[1]=y+20;
        body[2]=x+15;
        body[3]=y+20;
        body[4]=x+15;
        body[5]=y-20;
        body[6]=x-15;
        body[7]=y-20;
                //wheels
        setfillstyle(1,LIGHTBLUE);
        fillellipse(x+7,y+20+dely,7,7);
        fillellipse(x-7,y+20+dely,7,7);
                //body
        setfillstyle(1,GREEN);
        fillpoly(4,body);
                //head
        setfillstyle(1,MAGENTA);
        pieslice(x,y-22-dely,0,180,15);
                //eye
        setfillstyle(1,BLACK);
        char status[4];
        itoa(ehealth,status,10);
        outtextxy(x,y,status);
        if (dir==LEFT)     fillellipse(x-6,y-27-dely,1,2);
        else if (dir==RIGHT)       fillellipse(x+6,y-27-dely,1,2);
        else if (dir==DOWN) fillellipse(x,y-23-dely,2,2);
        else  fillellipse(x,y-27-dely,2,2);
        }

void erase(){
//cout<<x<<", "<<y<<endl;

        int body[8];
        setcolor(BLACK);

        body[0]=x-15;
        body[1]=y+20;
        body[2]=x+15;
        body[3]=y+20;
        body[4]=x+15;
        body[5]=y-20;
        body[6]=x-15;
        body[7]=y-20;
                //wheels
        setfillstyle(1,BLACK);
        fillellipse(x+7,y+20+dely,7,7);
        fillellipse(x-7,y+20+dely,7,7);
                //body
        setfillstyle(1,BLACK);
        fillpoly(4,body);
                //head
```

```
		setfillstyle(1,BLACK);
		pieslice(x,y-22-dely,0,180,15);
			//eye
		setfillstyle(1,BLACK);
		if (dir==LEFT)	fillellipse(x-5,y-25-dely,2,2);
		else if (dir==RIGHT)		fillellipse(x+6,y-27-dely,1,2);
		else fillellipse(x,y-23-dely,2,2);
		}

	void move(int dirin){
	erase();
	dely+=2;
		if (dely>4) dely=0;
	dir = dirin;
		switch (dirin) {
			case LEFT:
				x-=2;
				break;
			case RIGHT:
				x+=2;
				break;
			case UP:
				y-=2;
				break;
			case DOWN:
				y+=2;
				break;

		}
	draw();
	ex=x;
	ey=y;
	}

	void fire(){
 bulletE[bulletE_count].deploy(x,y,dir);
 bulletE_count++;
 if (bulletE_count==200) bulletE_count=0;
}


};

class r2d2:public character{
public:
	int dely;
	r2d2(int a, int b){
		dely=0;
		dir = RIGHT;
		x=a;
		y=b;
		phealth=10;
	}

	void draw(){
	//cout<<x<<", "<<y<<endl;
```

```
		int body[8];
		setcolor(BLACK);

		body[0]=x-15;
		body[1]=y+20;
		body[2]=x+15;
		body[3]=y+20;
		body[4]=x+15;
		body[5]=y-20;
		body[6]=x-15;
		body[7]=y-20;
			//wheels
		setfillstyle(1,WHITE);
		fillellipse(x+7,y+20+dely,7,7);
		fillellipse(x-7,y+20+dely,7,7);
			//body
		setfillstyle(1,RED);
		fillpoly(4,body);
			//head
		setfillstyle(1,YELLOW);
		pieslice(x,y-22-dely,0,180,15);
			//eye
		setfillstyle(1,BLACK);
		char status[4];
		itoa(phealth,status,10);
		outtextxy(x,y,status);
		if (dir==LEFT)	fillellipse(x-6,y-27-dely,1,2);
		else if (dir==RIGHT)		fillellipse(x+6,y-27-dely,1,2);
		else if (dir==DOWN) fillellipse(x,y-23-dely,2,2);
		else  fillellipse(x,y-27-dely,2,2);
		}

	void erase(){
	//cout<<x<<", "<<y<<endl;

		int body[8];
		setcolor(BLACK);

		body[0]=x-15;
		body[1]=y+20;
		body[2]=x+15;
		body[3]=y+20;
		body[4]=x+15;
		body[5]=y-20;
		body[6]=x-15;
		body[7]=y-20;
			//wheels
		setfillstyle(1,BLACK);
		fillellipse(x+7,y+20+dely,7,7);
		fillellipse(x-7,y+20+dely,7,7);
			//body
		setfillstyle(1,BLACK);
		fillpoly(4,body);
			//head
		setfillstyle(1,BLACK);
		pieslice(x,y-22-dely,0,180,15);
			//eye
```

```
				setfillstyle(1,BLACK);
				if (dir==LEFT)	fillellipse(x-5,y-25-dely,2,2);
				else if (dir==RIGHT)		fillellipse(x+6,y-27-dely,1,2);
				else fillellipse(x,y-23-dely,2,2);
				}

		void move(int dirin){
		erase();
		dely+=2;
				if (dely>4) dely=0;
		if (dirin!=3)		dir = dirin;
				switch (dirin) {
					case LEFT:
						x-=2;
						break;
					case RIGHT:
						x+=2;
						break;
					case UP:
						y-=2;
						break;
					case DOWN:
						y+=2;
						break;

				}
		draw();
		px=x;
		py=y;
		}

		void fire(){
 bullet[bullet_count].deploy(x,y,dir);
 bullet_count++;
 if (bullet_count==100) bullet_count=0;
}

};




int whichWay(char x){
//keyboard controls
				if (x=='w') return UP;
				if (x=='a') return LEFT;
				if (x=='s') return DOWN;
				if (x=='d') return RIGHT;

	//	cout<<"WRONG KEY "<<x;
		return 3;
		//getch();
}
int collisionEnemy(int bx,int by){

if (abs(bx-ex)<15)
		if (abs(by-ey)<23) ehealth--;
if (ehealth<=0) {
cout<<"\t\tYOU WIN";
getch();
exit(0);
}
if (abs(bx-ex)<15) if (abs(by-ey)<23) return 1;
}

int collisionPlayer(int bx,int by){
if (abs(bx-px)<15)
        if (abs(by-py)<23) phealth--;
if (phealth<=0) {
cout<<"\t\tYOU LOSE";
getch();
exit(0);
}
if (abs(bx-px)<15)
        if (abs(by-py)<23) return 1;
}

int main()              // test the functions
{
        cout<<"ROBO WARRIOR!!!\n\n\n\nWASD Movements.SPACE to fire. U to quit";
        getch();
  // initialize the graphics system
  int graphdriver = DETECT, graphmode;
  initgraph(&graphdriver, &graphmode, "..\\bgi");
  int movecount=0;
  int midx=getmaxx()/2;
  int midy=getmaxy()/2;
  r2d2 hanSolo(midx-100,midy);
  enemy darth_vader(midx+20,midy);
  char inp;
  int dirE=-1;
  clock_t world;
  world=clock();
  hanSolo.move(RIGHT);
  do {
  if (((clock()-world)/CLK_TCK)>0.01){
         for (int i=0;i<bullet_count;i++) bullet[i].draw();
         for (i=0;i<bulletE_count;i++) bulletE[i].draw();
         world=clock();
         movecount++;
         if (movecount>10) {
                darth_vader.fire();
                dirE=(rand()%4)-1;
                movecount=0;
         }
         darth_vader.move(dirE);
         if ((ex>getmaxx())||(ex<0)||(ey>getmaxy())||(ey<0)) darth_vader.move((dirE+1)%4-1);
         }

         if (kbhit()) {while(kbhit()) inp = getch();
                if (inp=='u') break;
                else if (inp==' ') {
                        hanSolo.fire();
                }
```

```
                    else hanSolo.move(whichWay(inp));
            }
            //cout<<hanSolo.x<<", "<<hanSolo.y;
  } while (1);
 closegraph();
  return 0;
}
```