ソフトゼミ B　第 3 回　自機・敵機の弾の描画と移動

■はじめに

今回みなさんにやってもらうことは、前回学習した自機と敵機の描画と移動の復習と応用とも言うべきものです。特に、敵の複数描画部分の復習が大幅に盛り込まれています。内容としてはタイトルに書いた通り、今度はシューティングゲームの核ともなる弾の描画と移動についてやっていきたいと思います。

■弾の初期化

まずはとにもかくにも、弾に関する情報の初期化をしていかなければいけません。弾の初期化は下のソースにあるように自機と敵機の弾で分けて書きます。初期化自体に関しては前回行った、敵の初期化と同じ方法で初期化をするので前回の復習だと思って、下のソースの穴埋め部分を埋めてみてください。 また、弾の初期化についてはできている人もいると思いますので、もしできている方は次の**自機と敵機の弾の発射**に進んでください。

```
//初期化用関数
int init(void)
{
        int i,j;
・・・・・・・・・・略・・・・・・・・・・・・・

        //player の玉の初期化
        for(j=0;j<【穴埋め】;j++)       {
                player_tama[j].x=0;
                player_tama[j].y=20;
                player_tama[j].life=0;
                player_tama[j].vx=0;
                player_tama[j].vy=20;
                player_tama[j].size_x=10;
                player_tama[j].size_y=10;
```

```
	}
	//敵と敵の弾の初期化
	for(i=0;i<ENEMY;i++)
	{
		enemy[i].life=0;
		enemy[i].size_x=50;
		enemy[i].size_y=50;
		enemy[i].vx=0;
		enemy[i].vy=10;
		enemy[i].x=0;
		enemy[i].y=0;
		for(j=0;j<【穴埋め】;j++)
		{
			enemy_tama[i][j].x=0;
			enemy_tama[i][j].y=0;
			enemy_tama[i][j].vx=0;
			enemy_tama[i][j].vy=20;
			enemy_tama[i][j].life=0;
			enemy_tama[i][j].size_x=5;
			enemy_tama[i][j].size_y=5;
		}
	}
・・・・・・・・・・略・・・・・・・・・・・・・
	return(1);
}
```

■自機と敵機の弾の発射

弾の初期化が終わった次にやることは、機体から弾を発射させることです。ここでは、その発射項目についてやっていきます。発射させるといっても、敵機は自動で弾が発射されるようにプログラムするので前回学習した敵の(配置)の応用です。次に自機ですが、プレイヤーが弾を自動で発射し続けるというのはつまらないので、ボタンを押すことによって弾が発射されるというようにしていきます(今回のゲームでは発射ボタンは一応**Z**ボタンを使用することにします)。このやり方も前回学習した、自機の移動の復習です。発射

さえしてしまえば、あとは敵機の弾とほとんど同じです。それでは**下のソースの穴埋め部分を埋めてみてください。**

```
//敵に弾を発射させる
void enemy_shot(void)
{
	int i,j;
	//一定フレームごとに
	if(time % 5 == 0)
	{
		//敵一体毎に（追加しました）
		for(i = 0;i <【穴埋め】;i++)
		{
			//敵の命がある時
			if(【穴埋め】>0)
			{
				for(j=0;j<【穴埋め】;j++)
				{
				//enemy_tama[i][j]の命があったらスキップ(追加しました)
					if(enemy_tama[i][j].life>0){【穴埋め】}
			//enemy_tama[i][j]の命があったらスキップ(追加しました)


				//enemy[i][j]の命がないとき創り出す
				enemy_tama[i][j].life=1;

				//敵の近くのいい感じのところに作る
			enemy_tama[i][j].x=enemy[i].x+enemy[i].size_x/2;
				enemy_tama[i][j].y=enemy[i].y+enemy[i].size_y;

		//これがないと1ケ所の座標が使用中でないものすべてに代入される
				break;
				}
			}
		}
	}
```

```
}


//player の弾を発射
void player_shot(void)
{
	int j;
	//キーボードの状態を取得
	GetHitKeyStateAll(keystate);
	//Z キーが押されていたら発射する。
	if(【穴埋め(ヒント: 上記の文章)】)
	{
		for(j=0;【穴埋め】;j++)
		{
			//弾の命があったらスキップ
			if(player_tama[j].life>0){【穴埋め】;}

			//弾の命がないとき創り出す
			player_tama[j].life=1;

			//player の近くのいい感じのところに作る

			player_tama[j].x=player.x+(rand()%((int)player.size_x/5))*5;
			player_tama[j].y=player.y;

			//これがないと1ケ所の座標が使用中でないものすべてに代入される
			break;
		}
	}

}
```

■自機と敵機の弾の移動

弾の発射ができたら次は移動です、移動に関しては敵機の移動の復習です。また、ゲームの処理速度を上げるために画面外に出てしまった弾は邪魔なので消去します。なのでこの判定を考えなければいけません、画面外に出たときの判定式が少し多くなっていますが、これは少し考えればわかるはずなので頑張って考えてみてください。それでは下のソースの穴埋め部分を埋めてみてください。

```
//敵の弾を動かす
void move_enemy_burret(void)
{
	int i,j;
	for(i=0;i<【穴埋め】;i++)
		for(j=0;j<【穴埋め】;j++)
		{
			//tama の命がある時動く
			if(【穴埋め】>0)
			{
				//座標を速度分動かす
				enemy_tama[i][j].x【穴埋め(ヒント:式)】;
				enemy_tama[i][j].y【穴埋め(ヒント:式)】;

				//画面外にでた弾を消す　x 軸にやったことを y 軸にも適応
				if(enemy_tama[i][j].x<0 ||
enemy_tama[i][j].x>WIDTH ||【穴埋め】 || 【穴埋め】)
				{
					////弾の HP をどうにかする.
					【穴埋め(ヒント:玉を消す式)】;
				}
			}
		}
}

//player の弾を動かす
void move_player_burret(void)
```

```
{
        int j;
                for(j=0;【穴埋め】;j++)
                {
                        //tama の命がある時動く
                        if(【穴埋め】)
                        {
                                //座標を速度分動かす
                                player_tama[j].x【穴埋め(ヒント:式)】;
                                player_tama[j].y【穴埋め(ヒント:式)】;
                        }

                        //画面外にでた弾を消す
                        if(player_tama[j].x < 0-player_tama[j].size_x ||
player_tama[j].x > WIDTH ||【穴埋め】 || 【穴埋め】)
                        {
                                【穴埋め(ヒント:玉を消す式)】;
                        }
                }
}
```

■自機と敵機の弾の描画

さて、弾の初期化(情報)に発射、そして移動と終わりました。最後に弾の描画をしていきましょう。今回穴埋めしてもらうのは、自分が書いた弾のグラフィックを読み込むための load 関数とそれを画面上に描画する draw 関数です。描画についても、前回やったことを思い出しながらやってもらえれば、穴を埋めることができると思います。それでは下のソースの穴埋め部分を埋めてみてください。

```
void load(void)
{
・・・・・・・・・・・・・略・・・・・・・・・・・・・
        //img フォルダの中から弾っぽいファイルを探してみる。
        imglist.player_tama=【穴埋め】;
        imglist.enemy_tama=【穴埋め】;
```

```
・・・・・・・・・・・・略・・・・・・・・・・・・・
}

void draw()
{
	int i,j;

・・・・・・・・・・・・略・・・・・・・・・・・・・
	for(i=0;i<ENEMY;i++)
	{
		//////敵の命がある時描画
		if(enemy[i].life>0)
		{
		DrawGraph((int)enemy[i].x,(int)enemy[i].y,imglist.enemy[i],TRUE);
		}
		for(j=0;j<【穴埋め】;j++)
		{
			//弾の命があるとき弾を描画
			if(【穴埋め】>0)
			{
				【穴埋め】;
			}
		}
	}
	for(j=0;j<【穴埋め】;j++)
	{
		//弾の命があるとき弾を描画
		if(【穴埋め】)
		{
			【穴埋め】;
		}
	}

・・・・・・・・・・・・・略・・・・・・・・・・・・・
	ScreenFlip();
}
```

■ゲームループ中の関数呼び出し

ここまでで、今回の自機と敵機の弾の描画に関する関数は全て完成しました。関数が完成したら、ゲーム中でその関数を使用するために関数呼び出しを行います。詳しいことは前回やったので、それを思い出しながらやってみてください。

それでは<u>下のソースの穴埋め部分を埋めてみてください。</u>

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nShowCmd ){

while(ProcessMessage()==0){

・・・・・・・・・・・・・略・・・・・・・・・・・・・
                while(ProcessMessage()==0){

                        //player の弾を発射する関数
                        【穴埋め】;
                        //player の弾を動かす
                        【穴埋め】;

                        //敵に弾を発射させる
                        【穴埋め】;
                        //敵の弾を動かす
                        【穴埋め】;

・・・・・・・・・・・・・略・・・・・・・・・・・・・
return 0;
}
```

■締め

本日の講座はここまでです。今回はほとんどが前回の復習でした、理解できていた人はつまらなかったかもしれませんがやはり復習というものは大事です。今回の復習でシューティングにおいて大きな部分を占める機体と弾の描画と移動についての理解を少しでも深めてもらえたら幸いです。さて、次回からこれもまたシューティングにおいて最も大事な当たり判定について学習していきますのでお楽しみに・・・。それではお疲れ様でした。