ソフトゼミ B 2015 第 3 回

# 自機・敵機の弾の描画と移動

## 1. はじめに

前回は自機及び敵機の描画と移動について学びました。今回は自機と敵機の弾の描画とその移動について学んでいきましょう。基本的な描画は前回の応用となります。

## 2. 弾の発射

前回のプログラム内にあった「playerbullet」「enemybullet」が自機の弾と敵機の弾の変数となります。それぞれ hp、vx、vy、atk、size_x、size_y が「int initialize」内で初期化されており、ここの数値を変更することでゲーム内での弾の動きが変化させることが可能です。なお、初期化用関数「int initialize」以外でも弾の動きや設定などを変更することが可能ですので興味がある方は挑戦してみてください。

### ▼ *自機からの発射*

まずは、自機から弾を発射していきます。プログラムに void create_playrebullet 関数を追加します。この関数は X ボタンを押すと自機の近くに弾を作り出す関数です。

弾の作り出される流れとしては、弾を発射するという命令（ここでは X ボタンを押すという行為）が出た場合、弾に HP を与え、x 座標、y 座標を設定するようになっています。弾の HP や座標の初期値に関しては初期化関数で設定したように 0 となっています。draw 関数を参考にしていただくとわかりますが、弾の体力がある時にのみ描画されるように設定されているため弾の HP が 0 の時は描画されません。

```
//自機の弾を作る関数
void create_playerbullet()
{
	static int t=0;　//X が連続で入力されていたフレーム数を格納,static だから初回だけ初期化されて、それからずっと値は保持される

	//X が入力されていたら
	if(【穴埋め】)
	{
		//連続入力フレーム数を 1 増加
		t++;
```

```
                //3 フレームごとに
                if(t%3==1)
                {
                        //すべての弾の中から
                        for(int i=0;i<BULLET;i++)
                        {
                                //存在していないものをさがし、弾を作る
                                if(playerbullet[i].【穴埋め】==0)
                                {
                                        //存在を与える
                                        playerbullet[i].【穴埋め】=1;
                                        //自機の正面に弾をつくる
                                        playerbullet[i].【穴埋め】 = player.x +
(player.size_x - playerbullet[i].size_x)/2;
                                        playerbullet[i].【穴埋め】 = player.y -
playerbullet[i].size_y;
                                        //1 フレームで 1 発だけ作るので、1 発作っ
たら break で抜ける
                                        break;
                                }
                        }
                }
        }
}
```

このプログラムでは 3 ループで 1 発の弾を撃つようになっています。つまり main 文内で while 文が 3 回ループする間に 1 発の弾が発射可能となっています。

### ▼ 敵機からの発射

次に敵機から弾を発射させます。今度は void create_enemyballte 関数を新たに追加してください。この関数は敵機の近くに弾を作り出す関数です。

```
//敵の弾を作る関数
void create_enemybullet()
{
	double ang;  //自機と敵の角度を格納する変数
	//すべての敵で
	for(int i=0;i<ENEMY;i++)
	{
		//存在しているやつ
		if(enemy[i].【穴埋め】!=0)
		{
			//その敵が作られてから何フレームかを格納している変数を
1 増加させる
			enemy[i].t++;
			//その敵が作られてから 15 フレーム周期ごとに
			if(enemy[i].t%15==0)
			{
				//その敵の弾で
				for(int j=0;j<BULLET;j++)
				{
					//存在していないものを探し,弾をつくる
					if(enemybullet[i][j].hp==0)
					{
						//存在を与え、(hp!=0 にする)
						enemybullet[i][j].【穴埋め】=1;
						//敵の正面に弾を作る
						enemybullet[i][j].【穴埋め】 =
enemy[i].x + enemy[i].size_x / 2;
						enemybullet[i][j].【穴埋め】 =
enemy[i].y + enemy[i].size_y;

						//自機狙いにしてみた
						//ang に自機と敵との角度を入れ
る
```

```
                                                    ang=atan2( (double) ( (player.y
+ player.size_y/2) -(enemy[i].y + enemy[i].size_y) ) , (double)( (player.x +
player.size_x/2) - (enemy[i].x + enemy[i].size_x / 2) ) );
                                                    enemybullet[i][j].vx          =
cos(ang)*8;
                                                    enemybullet[i][j].vy          =
sin(ang)*8;

                                                    //1 フレームで 1 発しか作りたく
ないので、1 発作ったら break で抜ける
                                                    break;
                                            }
                                    }
                            }
                    }
            }
}
```

敵機が存在しているときに 15 フレームに 1 発弾を発射します。15 フレームの判定は enemy[i].t で判定しており、これを 1 フレームに 1 づつ増やしていき 15 で割り切れる場合に弾を発射する処理をおこなっています。

enemybullet[i][j]は i 番めの敵の j 個目の弾という意味です。

## 3. 弾の移動

次に作り出した弾の移動を行います。

### ▼ 自機の弾の移動

この関数は弾を移動させるだけの関数です。前回のゼミで行った自機の移動と多少条件が違うだけでほぼ同じなのでそちらも参考にしてみて下さい。

```
//自機の弾を動かす関数
void move_playerbullet()
{
        //すべての弾(playerbullet[0]～playerbullet[BULLET-1(=199)])
```

```
for(int i=0;i<BULLET;i++)
{
	//hp!=0、すなわち存在していれば
	if(【穴埋め】)
	{
		//x,y 方向の速度を現座標に加える
		playerbullet[i].x +=【穴埋め】;
		playerbullet[i].y +=【穴埋め】;

		//y 方向で画面からいなくなったら、消す
		if(playerbullet[i].y < -playerbullet[i].size_y)
		{
			【穴埋め】=0;
		}
	}

	if(playerbullet[i].y < -playerbullet[i].size_y)
	{
		playerbullet[i].hp=0;
	}
	if(playerbullet[i].x < -playerbullet[i].size_x)
	{
		playerbullet[i].hp=0;
	}
}
```

### ▼ *敵機の弾の移動*

敵機の弾の移動は void move_enemybullet 関数で行っています。自機の弾の移動と同じため自分で解いてみて下さい。

## 4. ゲームループ中の関数呼び出し

ここまでに書いた関数もループ文中に組み込んで実行されるようにしなければ動きません。これも前回のゼミを参考に行って下さい。今回は数値の更新を行う関数 void update に上で作成した 4 つの関数を組み込んで下さい。