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

前回では自機と敵機の描画、そして移動についてしましたが今回はその自機・敵機から弾を発射させていこうと思います。基本的には前回の描画の応用となります。穴埋め問題は前回のプログラムを参考にして埋めてみてください。

◆ 弾の発射

前回のプログラムの中に「playerbullet の初期化」また「enemybullet の初期化」の文字があったと思います。ここに出てくる playerbullet また enemybullet はまさしく今回扱う敵機と自機の弾の変数です。それぞれ hp（弾の体力)、vx、vy（それぞれ x 軸また y 軸方向への移動速度)、atk（弾の攻撃力)、size_x、size_y（弾の大きさ、それぞれ横のサイズ縦のサイズ）が初期化用関数 int initialize 内にて初期化されています。ここの値を変えることでゲーム全体における弾の設定を変えることができます。被弾一回で食らうダメージを上げたい！や、弾の速度をもっと上げてみたい！と思ったらいじってみてください。ちなみに初期化用関数以外でももちろん弾の設定は変えれたりできます。弾の速度をだんだん早くするようにしたい、だんだん弾を大きくしていきたい、などと思った人は是非挑戦してみてください。

さて、本題に入ります。まず弾の発射についてやっていきたいと思います。

- **自機からの発射**

  次は自機からの発射をやりましょう。プログラムに自機の弾を作り出す関数 void create_playerbullet 関数を追加します。この関数では X ボタンが押された時自機の近くに弾を作り出すだけの関数です。

  弾の作り方としてはまず弾が発射されるという命令が来たとき（ここでは X ボタンが押された時)、弾の体力を 1 以上にする、そしてその弾の x 座標、y 座標を任意の場所に設定するという感じです。体力や座標の初期値は初期化関数で設定したとおり 0 になっています。描画の関数 draw を見ればわかりますが弾の体力があるときのみ描画されるように設定されているため、体力が 0 である場合は描画されません。

```
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】){
		t++;		//連続入力フレーム数を 1 増加
		//3 フレームごとに
		if(t%3==1){
			//すべての弾から
			for(inti=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){

					playerbullet[i].【穴埋め】=1; //存在を与え
					playerbullet[i].【穴埋め】=player.x+(player.size_x - playerbullet[i].size_x)/2;
 //自機の正面に弾を作る
					playerbullet[i].【穴埋め】=player.y-playerbullet[i].size_y;
					//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;
				}
			}
		}
	}else{ //X が入力されていなかったら
		t=0;  //連続入力フレーム数を 0 にする
	}
}
```

今回のプログラムでは 3 ループで一発の弾の発射、つまり main 文での while 文が 3 ループするうちに一発しか弾が発射できない仕様になっています。

- **敵機からの発射**

次に敵機からの発射をやっていきましょう。今回は前回のプログラムに新たに敵の弾を作り出す関数 void create_enemybullet 関数を追加します。この関数内では画面内に存在している敵の近くに弾を作り出すだけの関数です。

```
void create_enemybullet(){
        //すべての敵で
        for(inti=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].【穴埋め】==0){
                                                enemybullet[i][j].【穴埋め】=1;
                                                //存在を与え
                                                enemybullet[i][j].【穴埋め】 =
                                                enemy[i].x + enemy[i].size_x / 2;
                                                //敵の正面に弾を作る
                                                enemybullet[i][j].【穴埋め】 =
                                                enemy[i].y +enemy[i].size_y;
        //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;
                                        }
                                }
                        }
                }
        }
}
```

敵が存在しているとき、15 フレームごとに弾を発射します。15 フレームの判定は enemy[i].t で判定しています。これを 1 フレームで 1 ずつ増やしていき、15 で割り切れるようになったとき弾を発射します。

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

◆ 弾の移動

次に作り出した弾を移動させましょう。

- **自機の弾の移動**

この関数では作り出された弾を移動させるだけの変数です。

弾の移動とはどのように…？と思った方もいるかもしれませんが前回のゼミでやった自機の移動と同じです。移動する条件が違うだけだったりします。弾が存在する条件は上の弾の発射のプログラムが理解できたなら簡単だと思います。

```
//自機の弾を動かす関数
void move_playerbullet(){
	//すべての弾
	for(inti=0;i<BULLET;i++){
		//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、存在を消す
			if(playerbullet[i].y < -playerbullet[i].size_y){
				【穴埋め】=0;
			}
		}
	}
}
```

- **敵機の弾の移動**

敵機の弾の移動も自機の弾の移動と全く同じです。

敵機の弾の移動は void move_enemybullet 関数で行っています。同じなためここではプログラムを載せませんが解いてみてください。

さて、ここまで進めることで弾の発射、そして移動はできたと思います。次はこれを実行されるように組み込んでいきたいと思います。

◆ ゲームループ中の関数呼び出し

完成した関数をループ文中に組み込み実行されるようにされなきゃ意味がありません。
前回のゼミでのと同じなのでやってみてください。
今回は数値的な更新をする関数 void update に「自機の弾の発射の関数」「敵機の弾の発射の関数」「自機の弾の移動の関数」「敵機の弾の移動の関数」の 4 つを組み込んでみてください。プログラムはこの紙資料自体には載せないのでプログラムのコメントアウトを参考にしてください。

今回はちょっとした追加資料を用意しています。暇な人は挑戦してみてください。

◆ おしまい

さて、ついに弾が発射されるようになりました！しかし相手の撃った弾は自機をすり抜けてどこかに行っちゃいますね。その逆も然り。当たり判定がありません。次回では当たり判定を実装し弾を撃ったり撃たれたりを実装していきます。