My Changes / Notes of Interest
Matthew Barrera, Quake 2 Fall 2022

- Shortcut → Use “+set game modF22GoW”
- Config → Necessary for commands for GoW weapons
- Custom UI → Need pcx files to be in pak0 from baseq2
- I tried to document all changes made and I might have missed a few lines (fairly confident I didn’t); however, all of the changes did occur within these four files.

1. g_local.h
    a. Change some stats for the weapons & Make some notes for the weapons.

```
(a)
#define WEAP_SHOTGUN                    2 //Matthew B, NOTE: Use WEAP_SHOTGUN model for Leviathan Axe
#define WEAP_MACHINEGUN                 4 //Matthew B, NOTE: Use WEAP_MACHINEGUN model for Chaos Blades
#define WEAP_ROCKETLAUNCHER             8 //Matthew B, NOTE: Use WEAP_ROCKETLAUNCHER model for Talon Bow
#define WEAP_HYPERBLASTER               9 //Matthew B, NOTE: Use WEAP_BLASTER model for Draupnir Spear

//Matthew B, Create a Means of Death for each of my weapons, The actual dying can be similar to one of the others such as MOD_SHOTGUN
#define MOD_BLADES                      34      //Matthew B, Chaos Blades
#define MOD_AXE                         35      //Matthew B, Leviathan Axe
#define MOD_SPEAR                       36      //Matthew B, Draupnir Spear
#define MOD_BOW                         37      //Matthew B, Talon Bow

//Matthew B, The weapons have unique melee ranges, Experiment with the HSPREAD and VSPREAD values --> H is LeftRight and V is UpDown?
//Update -> Blades and Axe definitely use these values, Spear maybe not? I'm having an issue shortening the bullet range to mimic melee range.
#define DEFAULT_BLADES_HSPREAD      700     //Matthew B, Chaos Blades (LEFT to RIGHT)
#define DEFAULT_BLADES_VSPREAD      100     //Matthew B, Chaos Blades
#define DEFAULT_AXE_HSPREAD         100     //Matthew B, Leviathan Axe (TOP to BOTTOM)
#define DEFAULT_AXE_VSPREAD         1000    //Matthew B, Leviathan Axe
#define DEFAULT_SPEAR_HSPREAD       50      //Matthew B, Draupnir Spear (CLOSE to FAR)
#define DEFAULT_SPEAR_VSPREAD       50      //Matthew B, Draupnir Spear

//Matthew B, These bullet counts are large to avoid reloading issues, A sword doesn't need to be reloaded!
//Update -> Note sure how to create infinite ammo, so these values are more so set to maximum value we can allot them.
#define DEFAULT_DEATHMATCH_BLADES_COUNT 200     //Matthew B, Chaos Blades
#define DEFAULT_BLADES_COUNT            200     //Matthew B, Chaos Blades
#define DEFAULT_DEATHMATCH_AXE_COUNT    100     //Matthew B, Leviathan Axe
#define DEFAULT_AXE_COUNT               100     //Matthew B, Leviathan Axe
#define DEFAULT_DEATHMATCH_SPEAR_COUNT  200     //Matthew B, Draupnir Spear
#define DEFAULT_SPEAR_COUNT             200     //Matthew B, Draupnir Spear
#define DEFAULT_DEATHMATCH_BOW_COUNT    50      //Matthew B, Talon Bow
#define DEFAULT_BOW_COUNT               50      //Matthew B, Talon Bow

// powerup timers
float                   upgrade_framenum; //Matthew B

//Matthew B, NOTE!
//hyperblaster is spear   → Uses fire_blaster → Damage & SPEED & EFFECT & HYPER (KEEP MOD_SPEAR, Spear HSpread and VSpread might be useless but keep for now, Make hyper=FALSE)
```

```
//shotgun is axe            → Uses fire_bullet  → Damage & Kick & HSpread & VSpread & MOD_AXE
//machine gun is blades     → Uses fire_shotgun → Damage & Kick & HSpread & VSpread & MOD_BLADES
//rocket launcher is bow    → Uses fire_rocket  → Damage & SPEED & DamageRadius & RadiusDamage (KEEP MOD_BOW)
```

2. g_cmds.c
    a. Created a command. → Use “modhelp” to print the help screen.
    b. Utilized the give command → Use “give blades/axe/spear/bow” to give the corresponding weapon.

```
(a)
/*
==============
Cmd_ModHelp_f - Matthew B, the Help Screen
==============
*/
void Cmd_ModHelp_f(edict_t *ent)
void Cmd_ModHelp_f(edict_t *ent)
{
        Com_Printf("\n");   //This is to emulate a CLEAR command, only help text should be visible
        Com_Printf("God of War Mod by Matthew B, 2022:\n\n\n");
        Com_Printf("Spawn Axe:    give axe    (In command line)\n");
        Com_Printf("Spawn Spear:  give spear  (In command line)\n");
        Com_Printf("Spawn Blades: give blades (In command line)\n");
        Com_Printf("Spawn Bow:    give bow    (In command line)\n");
        Com_Printf("             No need for 'give ammo' because all weapons have infinite ammo.\n");
        Com_Printf("\n");
        Com_Printf("Rage Mode: Activated when HP drops to less than 50% of MAX HP.");
        Com_Printf("\n");
        Com_Printf("Unlock Weapon Upgrades: give weapon upgrade (In command line)\n");
        Com_Printf("                        Then, press 'r' to use the weapon upgrade powerup. --> Last 30 seconds.\n");
        Com_Printf("\n");
        Com_Printf("\n");
        Com_Printf("Scroll Wheel Up:           Use Axe\n");
        Com_Printf("Scroll Wheel Down:         Use Spear\n");
        Com_Printf("Scroll Wheel Press (MOUSE3): Use Blades\n");
        Com_Printf("Right-Click (MOUSE2):      Use Bow\n");
        Com_Printf("\n");
        Com_Printf("\n");
        Com_Printf("(1) Axe:           Left Click to Fire BIG Damage, but with LONG Cooldown. --> Top to Bottom Bullet Pattern.\n");
        Com_Printf("(2) Axe (Rage Mode): Remove the Long Cooldown & Deal BIG BIG DAMAGE.\n");
        Com_Printf("(3) Axe (Upgrade):   Alternate between NOCLIP & BOUNCE movetypes for 30 seconds.\n");
        Com_Printf("                     Afterwards, the next shot will return you back to WALK movetype. --> Stacks with Rage Mode effects.\n");
        Com_Printf("\n");
        Com_Printf("\n");
        Com_Printf("(4) Spear:         Left Click to Fire MID Damage, but with MID Cooldown. --> Front to Back Bullet Pattern.\n");
        Com_Printf("(5) Spear (Rage Mode): Change Particle Effects & Fire grenades in addition to the Spear Damage.\n");
        Com_Printf("(6) Spear (Upgrade):   Add New Particle Effects & Fire railgun in addition to the Spear Damage. --> Stacks with Rage Mode
effects.\n");
        Com_Printf("\n");
        Com_Printf("\n");
        Com_Printf("(7) Blades:          Left Click to Fire SML Damage, but with LOW Cooldown. --> Left to Right Bullet Pattern.\n");
        Com_Printf("(8) Blades (Rage Mode): Slight Increase to Damage & There is an Additional Blades Bullet Pattern (Top to Bottom).\n");
        Com_Printf("                        Heal +1 HP per Attack & Once above Rage Mode threshold Heal ALL Remaining HP.\n");
        Com_Printf("(9) Blades (Upgrade):   Heal +2 HP per Attack & Increase Max HP +1 per Attack for 30 seconds, with No Maximum value.\n");
        Com_Printf("\n");
        Com_Printf("\n");
        Com_Printf("(10) Bow:          Left Click to Fire BIG Damage, but with MID Cooldown. --> Front to Back Bullet Pattern.\n");
```

```
		Com_Printf("			Must directly Hit the Target. --> No More Splash Damage & It Costs HP to Fire Weapon (5 HP per Shot).\n");
		Com_Printf("(11) Bow (Rage Mode): Slight increase to damage & Gain +10 ARMOR per Attack, with Maximum value of 200.\n");
		Com_Printf("(12) Bow (Upgrade):   Fires Two Additional Rockets. --> One Traveling 1/3 slower, BIG Splash Damage. --> Second Traveling
1/10 slower, BIG BIG BIG Splash Damage.\n");
		return;
}

if (Q_stricmp(cmd, "modhelp") == 0)
{
		//Matthew B, This is the MODHELP command, essentially a help screen to tell the player how to use the mod
		Cmd_ModHelp_f (ent);
		return;
}
```

(b)

```
//Matthew B, This if statement can be used to give weapons
//Visit g_items.c for more information
//Essentially, we cycle through possible pick-up weapons
//Look for: "blades" | "axe" | "spear" | "bow"
```

3. g_items.c
    a. Create a powerup that will unlock the weapon upgrades temporarily.
    b. Create the weapons and finish the powerup via the item list. → We need to be able to “Give” them via command.
        i. We can also give the weapons new UI elements here.

(a)

```
//Matthew B, Custom Functions similar to Use_Quad
void Use_Weapon_Upgrade(edict_t* ent, gitem_t* item);
static int upgrade_drop_timeout_hack;

//Matthew B, My Weapon Upgrade powerup --> Unlock new abilities while the powerup is active
void Use_Weapon_Upgrade (edict_t* ent, gitem_t* item)
{
		int			timeout;

		ent->client->pers.inventory[ITEM_INDEX(item)]--;
		ValidateSelectedItem(ent);

		if (upgrade_drop_timeout_hack)
		{
			timeout = upgrade_drop_timeout_hack;
			upgrade_drop_timeout_hack = 0;
		}
		else
		{
			timeout = 300;
		}

		if (ent->client->upgrade_framenum > level.framenum)
			ent->client->upgrade_framenum += timeout;
		else
			ent->client->upgrade_framenum = level.framenum + timeout;

		gi.sound(ent, CHAN_ITEM, gi.soundindex("items/protect.wav"), 1, ATTN_NORM, 0);
}
```

(b)

```
//Matthew B, These are the new weapons we want to represent
void Weapon_Blades (edict_t* ent);
void Weapon_Spear (edict_t* ent);
void Weapon_Axe (edict_t* ent);
void Weapon_Bow (edict_t* ent);

//Matthew B, Don't forget hyperblaster = spear & shotgun = axe & machine gun = blades & rocket launcher = bow
//We will change the behavior of blaster from spear (for example), but we want to re-use the model and effects
//The icon for the UI has been changed to my own custom image
{
	//machine gun = blades (Heavy basis from that code)
	"weapon_blades", //The name of the weapon in the command line?
	Pickup_Weapon,
	Use_Weapon,
	Drop_Weapon,
	Weapon_Blades, //From p_weapon.c
	"misc/w_pkup.wav",
	"models/weapons/g_machn/tris.md2", EF_ROTATE,
	"models/weapons/v_machn/tris.md2",
	/* icon */		"w_blades",
	/* pickup */		"Blades",
				0,
				1,
				"Bullets",
				IT_WEAPON | IT_STAY_COOP,
				WEAP_MACHINEGUN,
				NULL,
				0,
				/* precache */ "weapons/machgf1b.wav weapons/machgf2b.wav weapons/machgf3b.wav weapons/machgf4b.wav
weapons/machgf5b.wav"
},
{
	//shotgun = axe (Heavy basis from that code)
	"weapon_axe", //The name of the weapon in the command line?
	Pickup_Weapon,
	Use_Weapon,
	Drop_Weapon,
	Weapon_Axe, //From p_weapon.c
	"misc/w_pkup.wav",
	"models/weapons/g_shotg/tris.md2", EF_ROTATE,
	"models/weapons/v_shotg/tris.md2",
	/* icon */		"w_axe",
	/* pickup */		"Axe",
				0,
				1,
				"Shells",
				IT_WEAPON | IT_STAY_COOP,
				WEAP_SHOTGUN,
				NULL,
				0,
				/* precache */ "weapons/shotgf1b.wav weapons/shotgr1b.wav"
},
{
	//hyperblaster = spear (Heavy basis from that code)
	"weapon_spear", //The name of the weapon in the command line?
	Pickup_Weapon,
	Use_Weapon,
	Drop_Weapon,
	Weapon_Spear, //From p_weapon.c
	"misc/w_pkup.wav",
	"models/weapons/g_hyperb/tris.md2", EF_ROTATE,
```

```
	"models/weapons/v_hyperb/tris.md2",
	/* icon */		"w_spear",
	/* pickup */	"Spear",
				0,
				1,
				"Cells",
				IT_WEAPON | IT_STAY_COOP,
				WEAP_HYPERBLASTER,
				NULL,
				0,
				/* precache */ "weapons/hyprbu1a.wav weapons/hyprbl1a.wav weapons/hyprbf1a.wav weapons/hyprbd1a.wav
misc/lasfly.wav"
},
{
	//rocket launcher = bow (Heavy basis from that code)
	"weapon_bow", //The name of the weapon in the command line?
	Pickup_Weapon,
	Use_Weapon,
	Drop_Weapon,
	Weapon_Bow, //From p_weapon.c
	"misc/w_pkup.wav",
	"models/weapons/g_rocket/tris.md2", EF_ROTATE,
	"models/weapons/v_rocket/tris.md2",
	/* icon */		"w_bow",
	/* pickup */	"Bow",
				0,
				1,
				"Rockets",
				IT_WEAPON | IT_STAY_COOP,
				WEAP_ROCKETLAUNCHER,
				NULL,
				0,
				/* precache */ "models/objects/rocket/tris.md2 weapons/rockfly.wav weapons/rocklf1a.wav weapons/rocklr1b.wav
models/objects/debris2/tris.md2"
},
{
	//Think about using item_ancient_head for level up stats --> special item that gives +2 to maximum health
	//This code is based off of item quad
	//Create an powerup called "Weapon Upgrade" that will unlock new abilities when used
	"item_weapon_upgrade",
	Pickup_Powerup,
	Use_Weapon_Upgrade,
	Drop_General,
	NULL,
	"items/pkup.wav",
	"models/items/c_head/tris.md2", EF_ROTATE,
	NULL,
	/* icon */		"i_fixme",
	/* pickup */	"Weapon Upgrade",
	/* width */			2,
		60,
		NULL,
		IT_POWERUP,
		0,
		NULL,
		0,
		/* precache */ "items/protect.wav items/protect2.wav items/protect4.wav"

},
```

4. p_weapon.c
    a. Create the weapon functions, especially those mentioned in g_items.c
        i. Shotgun / Axe (Left Click Ability) → Long line of pure damage with infinite usage (Ammo doesn't deplete).
            1. Scroll Wheel Up to Access
            2. Big DMG (6 with kick 0 → Shotgun uses 4 and 8)
            3. Long CD
            4. Short RANGE
            5. TOP to BOTTOM
        ii. Axe Rage
            1. Removes CD limitations → Becomes a rapid fire weapon
        iii. Axe Upgrade
            1. Grants NOCLIP and BOUNCE for limited time
        iv. Machine Gun / Blades (Left Click Ability) → Wide line of tiny damage with grabbing ability (Pull enemies towards you) and with infinite usage (Ammo doesn't deplete).
            1. Scroll Wheel Click to Access
            2. Small DMG (2 with kick -100 → Machine Gun uses 8 and 2)
            3. Short CD
            4. Mid RANGE
            5. LEFT to RIGHT
        v. Blades Rage
            1. Heal +1 HP for each bullet shot, Once back above Rage Mode threshold RESTORE ALL HP
            2. Slight increase to damage IN ADDITION TO addition fire pattern (Essentially Blades is called twice)
        vi. Blades Upgrade
            1. Increase MAX HP by +1 and Heal +2 HP for each bullet shot for a limited time
            2. Slight increase to damage
        vii. Hyper Blaster / Spear (Left Click Ability) → Straight line of damage with infinite usage (Ammo doesn't deplete).
            1. Scroll Wheel Down to Access
            2. Mid DMG (10 → Hyperblaster uses 20)
            3. Mid CD
            4. Long RANGE
            5. FRONT to BACK
        viii. Spear Rage
            1. Change the particle effects
            2. Fire grenades IN ADDITION TO the hyperblaster

ix. Spear Upgrade
    1. Add a new particle effects
    2. Fire railgun IN ADDITION to hyperblaster

x. Rocket Launcher / Bow (Left Click Ability) → Straight line of damage with infinite usage, but must directly hit the target (Damage Increased, Radius Damage Decreased) and it takes health to fire the weapon.
    1. Right Click to Access
    2. Big DMG (200 → Rocket Launcher uses 100+Random Value)
    3. Mid CD
    4. Infinite RANGE (Or rather, not a unique value anyway → All the new weapons have infinite range)
    5. FRONT to BACK

xi. Bow Rage
    1. Gain +10 Armor for each rocket shot, Once Armor values go above 200, drop is back down to 200 (Maximum cap)
    2. Slight increase to damage

xii. Rocket Upgrade
    1. Fires two additional rockets, one traveling 1/3 the speed of the original but BIG splash damage and the other traveling 1/10 the speed of the original but BIG BIG BIG splash damage

(a)

```
static qboolean is_upgrade; //Matthew B

is_upgrade = (ent->client->upgrade_framenum > level.framenum); //Matthew B

/*
=============================================================================

GOD OF WAR MOD WEAPONS - Matthew B

=============================================================================
*/

//Based off of weapon_shotgun_fire
void weapon_axe_fire(edict_t* ent)
{
	//Start, forward/right, offset --> Used for position.
	vec3_t		start;
	vec3_t		forward, right;
	vec3_t		offset;
	vec3_t		behind; //Matthew B, my own stat

	//Damage and kick --> Used for weapon attack value and knockback.
	int			damage = 6; //Based on shotgun, damage = 4
	int			kick = 0;   //Based on shotgun, kick = 8
```

```
// CHECK FOR WEAPON UPGRADE --- CONDITION IS IF ITEM IS IN INVENTORY
if (is_upgrade)
{
	if (ent->movetype == MOVETYPE_WALK || ent->movetype == MOVETYPE_BOUNCE)
	{
		ent->movetype = MOVETYPE_NOCLIP;
	}
	else
	{
		ent->movetype = MOVETYPE_BOUNCE;
	}
}
else
{
	ent->movetype = MOVETYPE_WALK;
}


//CHECK FOR RAGE MODE --- CONDITION IS IF HP < 50%
if (ent->health < (ent->max_health) / 2)
{
	if (ent->client->ps.gunframe == 2)
	{
		ent->client->ps.gunframe++;
		return;
	}
}
else {
	if (ent->client->ps.gunframe == 9)
	{
		ent->client->ps.gunframe++;
		return;
	}
}

//q_shared.c --> (angles, forward, right, up)
AngleVectors(ent->client->v_angle, forward, right, NULL);

//q_shared.c --> (in, scale, out)
VectorScale(forward, -2, ent->client->kick_origin);

ent->client->kick_angles[0] = -2;

//q_shared.h --> (v, x, y, z) --> (v[0]=(x), v[1]=(y), v[2]=(z))
VectorSet(offset, 0, 8, ent->viewheight - 8); //Originally was (offset, 0, 8, ent->viewheight - 8)

//p_weapon.c --> (client, point, distance, forward, right, result) -- distance copied into _distance
//g_util.c --> G_ProjectSource --> (point, distance, forward, right, result) --> result[0] [1] [2]
P_ProjectSource(ent->client, ent->s.origin, offset, forward, right, start);
//Matthew B, IDEA: Change value of distance (ent->s.origin) to get limited range

//Quad damage modifier
if (is_quad)
{
	damage *= 4;
	kick *= 4;
}


//Deathmatch modifier
```

```
	//g_weapon.c --> (self, start, aimdir, damage, kick, hspread, vspread, count, mod) --> For i = 0; i < count; i++ {Shoot Bullets via fire_lead}
	//gw_weapon.c --> fire_lead --> (self, start, aimdir, damage, kick, te_impact, hspread, vspread, mod)
	//Leviathan Axe:
	// o Relatively LONG CD and BIG DMG and SHORT RANGE

	if (deathmatch->value)
		fire_shotgun(ent, start, forward, damage, kick, DEFAULT_AXE_HSPREAD, DEFAULT_AXE_VSPREAD,
DEFAULT_DEATHMATCH_AXE_COUNT, MOD_AXE); //Previously was 500/500 for H and V Spread (From shotgun)
	else
		VectorScale(forward, -1, behind);
		fire_shotgun(ent, start, forward, damage, kick, DEFAULT_AXE_HSPREAD, DEFAULT_AXE_VSPREAD,
DEFAULT_AXE_COUNT, MOD_AXE); //Fires the gun forward --> Previously was 500/500 for H and V Spread (From shotgun)
		//fire_shotgun(ent, start, behind, damage, kick, DEFAULT_AXE_HSPREAD, DEFAULT_AXE_VSPREAD,
DEFAULT_AXE_COUNT, MOD_AXE);  //Fires the gun behind --> Previously was 500/500 for H and V Spread (From shotgun)

	// send muzzle flash
	gi.WriteByte(svc_muzzleflash);
	gi.WriteShort(ent - g_edicts);
	gi.WriteByte(MZ_SHOTGUN | is_silenced);
	gi.multicast(ent->s.origin, MULTICAST_PVS);

	ent->client->ps.gunframe++;
	PlayerNoise(ent, start, PNOISE_WEAPON);


	//Matthew B, we can get infinite ammo by removing the condition to deplete ammo
	if (false)
	//if (!((int)dmflags->value & DF_INFINITE_AMMO)) --> Original if statement
		ent->client->pers.inventory[ent->client->ammo_index]--;
}

//Based off of Weapon_Shotgun
void Weapon_Axe(edict_t* ent)
{

	static int pause_frames[] = { 22, 28, 34, 0 };
	static int fire_frames[] = { 20, 21, 0 }; //Originally 8,9,0

	//CHECK FOR RAGE MODE --- CONDITION IS IF HP < 50%
	if (ent->health < (ent->max_health) / 2)
	{
		//Change the rate of fire
		static int pause_frames[] = { 7, 8, 10, 0 }; //Orignally 22,28,34,9
		static int fire_frames[] = { 2, 3, 0 }; //Originally 8,9,0

		Weapon_Generic(ent, 1, 4, 11, 13, pause_frames, fire_frames, weapon_axe_fire);
	}
	else {
		//The main Weapon_Generic WITHOUT the RAGE MODE
		Weapon_Generic(ent, 1, 19, 36, 39, pause_frames, fire_frames, weapon_axe_fire); //Originally 7,18,36,39
	}                                                                   //int FRAME_ACTIVATE_LAST, int FRAME_FIRE_LAST, int FRAME_IDLE_LAST, int
FRAME_DEACTIVATE_LAST
}




//Based off of Machinegun_Fire
void Blades_Fire(edict_t* ent)
{
	int	i;
```

```
//Start, forward/right, angles --> Used for position.
vec3_t		start;
vec3_t		forward, right;
vec3_t		angles;

//Damage and kick --> Used for weapon attack value and knockback.
int			damage = 2; //Based on machine gun, damage = 8
int			kick = -100;		//Based on machine gun, damage = 2

//Offset --> Used for position.
vec3_t		offset;

// CHECK FOR WEAPON UPGRADE --- CONDITION IS IF ITEM IS IN INVENTORY
if (is_upgrade)
{
	ent->max_health += 1;
	ent->health += 2;
	damage *= 2; //Slight buff to damage again
}

if (!(ent->client->buttons & BUTTON_ATTACK))
{
	ent->client->machinegun_shots = 0;
	ent->client->ps.gunframe++;
	return;
}

if (ent->client->ps.gunframe == 5)
	ent->client->ps.gunframe = 4;
else
	ent->client->ps.gunframe = 5;

if (ent->client->pers.inventory[ent->client->ammo_index] < 1)
{
	ent->client->ps.gunframe = 6;
	if (level.time >= ent->pain_debounce_time)
	{
		gi.sound(ent, CHAN_VOICE, gi.soundindex("weapons/noammo.wav"), 1, ATTN_NORM, 0);
		ent->pain_debounce_time = level.time + 1;
	}
	NoAmmoWeaponChange(ent);
	return;
}

if (is_quad)
{
	damage *= 4;
	kick *= 4;
}

for (i = 1; i < 3; i++)
{
	ent->client->kick_origin[i] = 0; //Was originally crandom() * 0.35
	ent->client->kick_angles[i] = 0; //Was originally crandom() * x0.70
}
ent->client->kick_origin[0] = 0; //Was originally crandom() * 0.35
ent->client->kick_angles[0] = ent->client->machinegun_shots * -1.5;

// raise the gun as it is firing
if (!deathmatch->value)
{
```

```
		ent->client->machinegun_shots++;
		if (ent->client->machinegun_shots > 0) //Was originally >9 --> We want to remove all recoil
			ent->client->machinegun_shots = 0; //Was originall = 9
	}


	// get start / end positions
	VectorAdd(ent->client->v_angle, ent->client->kick_angles, angles);
	AngleVectors(angles, forward, right, NULL);
	VectorSet(offset, 0, 8, ent->viewheight - 8);
	P_ProjectSource(ent->client, ent->s.origin, offset, forward, right, start);


	//CHECK FOR RAGE MODE --- CONDITION IS IF HP < 50%
	if (ent->health < (ent->max_health) / 2)
	{
		ent->health += 1;
		if (ent->health >= (ent->max_health) / 2)
		{
			ent->health += (ent->max_health) / 2;
		}
		damage *= 2; //Slight buff to damage
		fire_bullet(ent, start, forward, damage, kick, DEFAULT_BLADES_VSPREAD, DEFAULT_BLADES_HSPREAD, MOD_BLADES);
//Adds another layer of bullet shots, inverted H and V spread
	}


	fire_bullet(ent, start, forward, damage, kick, DEFAULT_BLADES_HSPREAD, DEFAULT_BLADES_VSPREAD, MOD_BLADES);

	gi.WriteByte(svc_muzzleflash);
	gi.WriteShort(ent - g_edicts);
	gi.WriteByte(MZ_MACHINEGUN | is_silenced);
	gi.multicast(ent->s.origin, MULTICAST_PVS);

	PlayerNoise(ent, start, PNOISE_WEAPON);

	//Matthew B, we can get infinite ammo by removing the condition to deplete ammo
	if (false)
	//if (!((int)dmflags->value & DF_INFINITE_AMMO)) --> Original if statement
		ent->client->pers.inventory[ent->client->ammo_index]--;

	ent->client->anim_priority = ANIM_ATTACK;
	if (ent->client->ps.pmove.pm_flags & PMF_DUCKED)
	{
		ent->s.frame = FRAME_crattak1 - (int)(random() + 0.25);
		ent->client->anim_end = FRAME_crattak9;
	}
	else
	{
		ent->s.frame = FRAME_attack1 - (int)(random() + 0.25);
		ent->client->anim_end = FRAME_attack8;
	}
}

//Based off of Weapon_Machinegun
void Weapon_Blades(edict_t* ent)
{
	static int pause_frames[] = { 6, 7, 0 }; //Originally 23, 45, 0
	static int fire_frames[] = { 4, 5, 0 }; //Originally 4, 5, 0

	Weapon_Generic(ent, 3, 5, 7, 8, pause_frames, fire_frames, Blades_Fire); //Originally 3, 5, 45, 49
```

```
                                                        //int FRAME_ACTIVATE_LAST, int FRAME_FIRE_LAST, int FRAME_IDLE_LAST, int
FRAME_DEACTIVATE_LAST
}



//Based off of Blaster_Fire
void Spear_Fire(edict_t* ent, vec3_t g_offset, int damage, qboolean hyper, int effect)
{
	//Start, forward/right, offset --> Used for position.
	vec3_t	forward, right;
	vec3_t	start;
	vec3_t	offset;

	if (is_quad)
		damage *= 4;
	AngleVectors(ent->client->v_angle, forward, right, NULL);
	VectorSet(offset, 24, 8, ent->viewheight - 8);
	VectorAdd(offset, g_offset, offset);
	P_ProjectSource(ent->client, ent->s.origin, offset, forward, right, start);

	VectorScale(forward, -2, ent->client->kick_origin);

	ent->client->kick_angles[0] = -1;

	effect = EF_GIB;
	//CHECK FOR RAGE MODE --- CONDITION IS IF HP < 50%
	if (ent->health < (ent->max_health) / 2)
	{
		effect = EF_GREENGIB;
		fire_grenade(ent, start, forward, 50, 2000, 0.5, 80); // *self, start, aimdir, damage, speed, timer, damage_radius
	}

	// CHECK FOR WEAPON UPGRADE --- CONDITION IS IF ITEM IS IN INVENTORY
	if (is_upgrade)
	{
		fire_rail(ent, start, forward, 50, 100); //void fire_rail (edict_t *self, vec3_t start, vec3_t aimdir, int damage, int kick);
	}

	hyper = 0; //To ensure that hyperblaster functions has more blaster behaviour instead
	fire_blaster(ent, start, forward, 10, 2000, effect, hyper); //Matthew B, was originally (ent, start, forward, DAMAGE, 1000, EFFECT, HYPER)
--> EFFECT = EF_HYPERBLASTER

	// send muzzle flash
	gi.WriteByte(svc_muzzleflash);
	gi.WriteShort(ent - g_edicts);
	if (hyper)
		gi.WriteByte(MZ_HYPERBLASTER | is_silenced);
	else
		gi.WriteByte(MZ_BLASTER | is_silenced);
	gi.multicast(ent->s.origin, MULTICAST_PVS);

	PlayerNoise(ent, start, PNOISE_WEAPON);
}

//Based off of Weapon_HyperBlaster_Fire
void Weapon_Spear_Fire(edict_t* ent)
{
	float	rotation;
	vec3_t	offset;
```

```
	int		effect;
	int		damage;

	ent->client->weapon_sound = gi.soundindex("weapons/hyprbl1a.wav");

	if (!(ent->client->buttons & BUTTON_ATTACK))
	{
		ent->client->ps.gunframe++;
	}
	else
	{
		if (!ent->client->pers.inventory[ent->client->ammo_index])
		{
			if (level.time >= ent->pain_debounce_time)
			{
				gi.sound(ent, CHAN_VOICE, gi.soundindex("weapons/noammo.wav"), 1, ATTN_NORM, 0);
				ent->pain_debounce_time = level.time + 1;
			}
			NoAmmoWeaponChange(ent);
		}
		else
		{
			rotation = (ent->client->ps.gunframe - 5) * 2 * M_PI / 3; //Rotation originally divded by 6
			offset[0] = sin(rotation); //Originally times by -4
			offset[1] = 0;
			offset[2] = cos(rotation); //Originally times by 4

			if ((ent->client->ps.gunframe == 6) || (ent->client->ps.gunframe == 9))
				effect = EF_HYPERBLASTER;
			else
				effect = 0;
			if (deathmatch->value)
				damage = 15;
			else
				damage = 20;
			Spear_Fire(ent, offset, damage, true, effect);
			//Matthew B, we can get infinite ammo by removing the condition to deplete ammo
			if(false)
			//if (!((int)dmflags->value & DF_INFINITE_AMMO)) --> Original if statement
				ent->client->pers.inventory[ent->client->ammo_index]--;

			ent->client->anim_priority = ANIM_ATTACK;
			if (ent->client->ps.pmove.pm_flags & PMF_DUCKED)
			{
				ent->s.frame = FRAME_crattak1 - 1;
				ent->client->anim_end = FRAME_crattak9;
			}
			else
			{
				ent->s.frame = FRAME_attack1 - 1;
				ent->client->anim_end = FRAME_attack8;
			}
		}

		ent->client->ps.gunframe++;
		if (ent->client->ps.gunframe == 12 && ent->client->pers.inventory[ent->client->ammo_index])
			ent->client->ps.gunframe = 6;
	}

	if (ent->client->ps.gunframe == 22) //Used to be 12
	{
```

```
                gi.sound(ent, CHAN_AUTO, gi.soundindex("weapons/hyprbd1a.wav"), 1, ATTN_NORM, 0);
                ent->client->weapon_sound = 0;
        }

}

//Based off of Weapon_HyperBlaster
void Weapon_Spear(edict_t* ent)
{
        static int pause_frames[] = { 0 };
        static int fire_frames[] = { 16, 17, 18, 19, 20, 21, 0 };  //Originally 6, 7, 8, 9, 10, 11, 0

        Weapon_Generic(ent, 15, 30, 49, 53, pause_frames, fire_frames, Weapon_Spear_Fire); //Originally 5, 20, 49, 53
                                                                        //int FRAME_ACTIVATE_LAST, int FRAME_FIRE_LAST, int FRAME_IDLE_LAST, int
FRAME_DEACTIVATE_LAST
}




//Based off of Weapon_RocketLauncher_Fire
void Weapon_Bow_Fire(edict_t* ent)
{
        //Start, forward/right, offset --> Used for position.
        vec3_t  offset, start;
        vec3_t  forward, right;

        int             damage;
        float   damage_radius;
        int             radius_damage;

        damage = 200; //Initially was 100 + (int)(random() * 20.0)
        radius_damage = 10;
        damage_radius = 10;
        if (is_quad)
        {
                damage *= 4;
                radius_damage *= 4;
        }

        AngleVectors(ent->client->v_angle, forward, right, NULL);

        VectorScale(forward, -2, ent->client->kick_origin);
        ent->client->kick_angles[0] = -1;

        VectorSet(offset, 8, 8, ent->viewheight - 8);
        P_ProjectSource(ent->client, ent->s.origin, offset, forward, right, start);

        //CHECK FOR RAGE MODE --- CONDITION IS IF HP < 50%
        if (ent->health < (ent->max_health) / 2)
        {
                gitem_t* it;
                gitem_armor_t* info;

                it = FindItem("Body Armor");
                info = (gitem_armor_t*)it->info;
                ent->client->pers.inventory[ITEM_INDEX(it)] += 10;
                damage *= 2;
                if (ent->client->pers.inventory[ITEM_INDEX(it)] >= 200)
                {
                        //A cap to prevent armor from skyrocketing to insane values
```

```
                                ent->client->pers.inventory[ITEM_INDEX(it)] = 200;

                        }
                }

                fire_rocket(ent, start, forward, damage, 660, damage_radius, radius_damage);

                // CHECK FOR WEAPON UPGRADE --- CONDITION IS IF ITEM IS IN INVENTORY
                if (is_upgrade)
                {
                        //Upgrade disables the health penalty
                        fire_rocket(ent, start, forward, damage, 220, 100, 200);
                        fire_rocket(ent, start, forward, damage, 66, 200, 2000);
                }
                else {
                        ent->health -= 5; //Matthew B, Take damage from firing this weapon; Prevents armor spam.
                        if (random() * 2 > 1)
                        {
                                gi.sound(ent, CHAN_VOICE, gi.soundindex("gunner/gunpain2.wav"), 1, ATTN_NORM, 0);
                        }
                        else
                        {
                                gi.sound(ent, CHAN_VOICE, gi.soundindex("gunner/gunpain1.wav"), 1, ATTN_NORM, 0);
                        }
                }

                // send muzzle flash
                gi.WriteByte(svc_muzzleflash);
                gi.WriteShort(ent - g_edicts);
                gi.WriteByte(MZ_ROCKET | is_silenced);
                gi.multicast(ent->s.origin, MULTICAST_PVS);

                ent->client->ps.gunframe++;

                PlayerNoise(ent, start, PNOISE_WEAPON);

                //Matthew B, we can get infinite ammo by removing the condition to deplete ammo
                if (false)
                        //if (!((int)dmflags->value & DF_INFINITE_AMMO)) --> Original if statement
                        ent->client->pers.inventory[ent->client->ammo_index]--;
}

//Based off of Weapon_RocketLauncher
void Weapon_Bow(edict_t* ent)
{
        static int pause_frames[] = { 19, 24, 42, 50, 0 }; //was originally 25, 33, 42, 59, 0
        static int fire_frames[] = { 8, 0 }; //was originally 5, 0

        Weapon_Generic(ent, 7, 17, 50, 54, pause_frames, fire_frames, Weapon_Bow_Fire); //was originally 4, 12, 50, 54
}
```