This document has been provided to the Modern Warfare Cubed team as is and is kept for archival purposes, no changes have been made to it, which make it potentially inaccurate for Modern Warfare Cubed.

Click here to see the official Modern Warfare Cubed docs

Vic's Modern Warfare 3.0
Content Guide

Adding new reticles

In the 'advanced warfare' package you will find "Reticles.java," which is where all the reticles go. All reticle textures should be added under the "crosshairs" folder. For holographic reticles, they are added like so:

```
/*
 * HOLOGRAPHIC RETICLES
 */

public static final Reticle HOLO_ONE = new Reticle("holo", 0.1f);
public static final Reticle OKP = new Reticle("okp", 0.1f);
```

Texture Name

Texture Scale

For sniper reticles, they are added the same, but without the "texture scale" argument as textures will always fill the scope.

```
/*
 * SNIPER RETICLES
 */

public static final Reticle SNIPER_RETICLE_ONE = new Reticle("reticle1");
```

Assigning reticles to holographic sights

All scopes are made in Attachments.java. Holographic sights follow this format:

```
Reflex = new ItemScope.Builder()
        .withHolographicReticles(Reticles.HOLO_ONE)
        .withReticlePositioning((model, itemStack) -> {
            GlStateManager.translate(-0.08, -3.15, -0.0);
            GlStateManager.scale(1.5, 1.5, 1.2);
        })
```

Reticles

Positioning

Reticles is a list of reticles, so you can add multiple, like so:

```
.withHolographicReticles(Reticles.HOLO_ONE, Reticles.OKP)
```

In addition, there is an optional additional parameter made for scopes like the OKP-7, where they are circular called "withRadialCut".

```
.withHolographicReticles(Reticles.HOLO_ONE, Reticles.OKP)
.withRadialCut(1.0f)
.withReticlePositioning((model, itemStack) -> {
    GlStateManager.translate(-0.08, -3.15, -0.0);
    GlStateManager.scale(1.5, 1.5, 1.2);
})
```

This parameter will make the holographic screen take on a circular shape. Another important thing to note is that the positioning does follow a live update, so if you make chances (while in debug mode in Eclipse, of course) it will reflect immediately in game.

Assigning reticles to sniper scopes

Sniper scopes are set up largely the same way– except you use a method called "withSniperReticle" which only accepts a single reticle, like so:

```
ACOG = new ItemScope.Builder()
        .withSniperReticle(Reticles.SNIPER_RETICLE_ONE)
        .withOpticalZoom().withZoomRange(0.22f, 0.1f)
        .withViewfinderPositioning((p, s) -> {
            GL11.glScalef(1.18f, 1.18f, 1.18f);
            GL11.glTranslatef(0.086f, 0.42f, 0.56f);
        })
```

Assigning magazine rotation points

```
Magazines.AK47PMAGTan = new ItemMagazine.Builder()
        .withRotationPoint(-0.20000000596046452, 0.44000001311302195, -2.20000006556511)
        .withAmmo(30).withCompatibleBullet(Bullets.Bullet762x39).withName("AK47PMAGTan")
        .withModId(ModernWarfareMod.MODID).withCreativeTab(ModernWarfareMod.AmmoTab)
```

Naming conventions for animations

If you want your animations to be set up properly in game, they need to be exported properly from BlockBench. The following are the names that must be set for each animation:

Animation	Technical Name
Load	load
Unload	unload
Load Empty	loadempty
Compound Reload*	reload
Compound Reload Empty*	reloadempty
Tactical Reload*	reloadtactical

Draw	draw
Inspect	inspect
Eject Spent Round	ejectspentround
Eject Spent Round Aimed	ejectspentroundaimed

**Supports two-magazine functionality (also known as "tactical reloading functionality")*

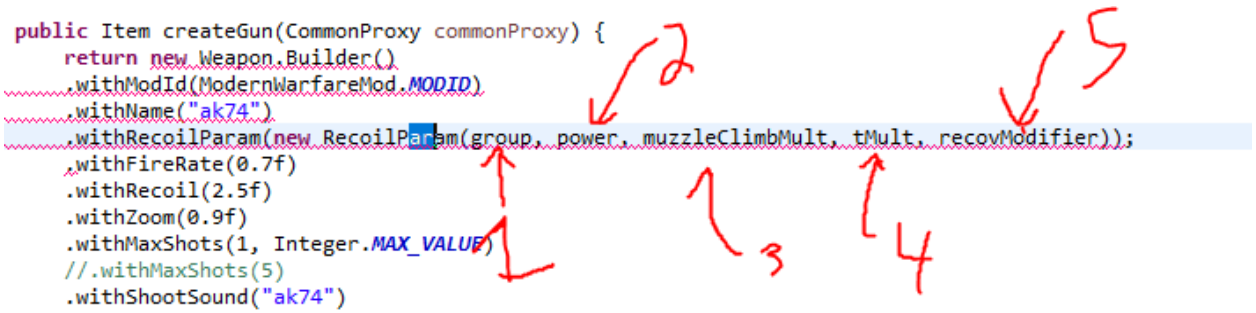
The filename is also important, it should look like the following:

```
> name.animation.json
  ↑
Animation Name
```

Naming conventions for parts

Part	Technical Name
Weapon	main
Left Hand	lefthand
Right Hand	righthand
Magazine	magazine
Extra Magazine	magazine_extra
Slide **or** Action	action

Recoil Parameters



1. Recoil Group: 0 for Assault Rifle type recoil, 1 for Pistol type recoil, 2 for Shotgun/Sniper/Other type
2. General power of the animation, standard is about 50
3. How much the gun rotates up

4. Translation on recoil, should look like this:

```
Mult, new Vec3d(x, y, z), recov
```

5. Should be a very small value (i.e. 0.05), changes how fast the gun recovers from the initial impulse of the recoil

Beizer

```
            GL11.glScaled(1F, 1F, 1F);
        })
        .withTextureNames("m4a1")
        .withRenderer(new WeaponRenderer.Builder()
            .withModId(ModernWarfareMod.MODID)
            .withModel(new M4A1())
            .withADSBeizer(new Vec3d(x, y, z))

            .withEntityPositioning(itemStack -> {
                GL11.glScaled(0.5F, 0.5F, 0.5F);
                GL11.glTranslatef(0, 0f, 3f);
```

Shell Type

```
    .withMuzzlePosition(new Vec3d(-0.1, -1.
    .withRecoil(1.5f)
    .withShellType(Type.ASSAULT)
    .withZoom(0.8f)
    .withMaxShots(1, Integer.MAX_VALUE)
    .withShootSound("m4a1")
    .withSilencedShootSound("m4a1_silenced"
```

Slide Recoil

```
.withTextureNames("mp40")
.withRenderer(new WeaponRenderer.Builder()
    .withModId(ModernWarfareMod.MODID)
    .withModel(new MP40())
    .withActionPiece(AuxiliaryAttachments.MP40action)
    .withActionTransform(new Transform().withPosition(0, 0, 0.8))
```

.withActionPiece specifies all the slides that this will work with. It can accept multiple attachments (i.e. for the glock you would want to include the razorback slide). .withActionTransform specifies the transform position.

You can mess around with the action position in game by typing:
/wdb weapon slide edit
Then
/wdb weapon slide setpos 0.0 0.0 0.0
Where the three numbers are x, y, and z respectively. You can then input those numbers into the transform's position argument in order to set its position.

Screen Shake

```
public Item createGun(CommonProxy commonProxy) {
    return new Weapon.Builder()
    .withModId(ModernWarfareMod.MODID)
    .withName("mp40")
    .withFireRate(0.55f)
    .withRecoil(1.5f)
    .withZoom(0.9f)
    .withModernScreenShaking(1000.0, 1.0)
    .withMaxShots(1, Integer.MAX_VALUE)
    //.withMaxShots(5)
```

To edit the screen shake, /wdb weapon shake edit then /wdb weapon shake set 0.0 0.0

Muzzle Flash

```
)lic Item createGun(CommonProxy commonProxy) {
  return new Weapon.Builder()
  .withModId(ModernWarfareMod.MODID)
  .withName("m17")
  .withFireRate(0.5f)
  .withRecoil(3f)
  .withZoom(0.9f)
  .withMuzzlePosition(new Vec3d(-0.2000000059604644, -1.0, -5.691999975919736)
  .withMaxShots(1)
  .withShootSound("m17")
  .withSilencedShootSound("colt_m45a1_silenced")
  .withReloadSound("m17_reload")
  .withUnloadSound("m17_unload")
  .withInspectSound("inspection")
  .withDrawSound("m17_draw")
  .withReloadingTime(50)
  .withCrosshair("gun")
  .withCrosshairRunning("Running")
  .withCrosshairZoomed("Sight")
```

Regenerating the Weapon Icon Sheet

In order to optimize loading, a weapon icon sheet is generated. In order to regenerate the weapon icon sheet, you first need to turn on liverender (/wdb weapon liverender toggle). You then need to go into the inventory and load *all the guns*. When you are done type (/wdb weapon buildsheet) and it will export guniconsheet.png to your run folder. You then put it in textures/gui/

Moving the Modification GUI for a Weapon

In game, type /wdb weapon gui and it will toggle GUI rearrangement, drag the GUI piece where you want and then type /wdb weapon gui print and it will export an array to console. In the weapon factory, you can then use .withGUIPositions([insert array here]) to set them.

Skin

GunSkins.java

Textures go in the textures/models folder

```
 public static ItemSkin Example;

    public static void init(Object mod, ConfigurationManager
configurationManager, CompatibleFmlPreInitializationEvent event) {

       GunSkins.Example = new
ItemSkin.Builder().withTextureVariant([textureName]).withModId(ModernWarfar
eMod.MODID)
```

```
.withCreativeTab(ModernWarfareMod.AttachmentsTab).withName("ExampleSkin")
                .build(ModernWarfareMod.MOD_CONTEXT, ItemSkin.class);
 CommonRegistry.gunSkins.add(GunSkins.Example);
    }
```

Registering sounds in the universal registry

In advanced warfare, you will find a class called ‘UniversalSoundRegistry.’ Within the init() method, you may register your sounds as follows:

```java
/**
 *
 * Creates a sound registry for WeaponLib to access in order to play
 * reloading noises.
 *
 * @author Jim Holden, 2022
 *
 */
public class UniversalSoundRegistry {


    public static void init() {
        // Example: UniversalSoundLookup.registerSoundToLookup("test_sound");
        UniversalSoundLookup.registerSoundToLookup("acr");
        UniversalSoundLookup.registerSoundToLookup("soos");

    }


}
```

Scopes using white phosphor

Use the line withWhitePhosphor() below hasNightVision().

Adding gun recipes

Use the line .withModernRecipe(ItemStack…stacks) which accepts a set of itemstacks as the parameter. For example, should I want the gun to be crafted with four gunmetal plates and four polymer composite plates, then I could do:

```
.withModernRecipe(new ItemStack(Ores.GunMetalPlate, 4), new
ItemStack(Ores.PolymerCompositePlate, 4));
```

Adding items used for crafting

In the old system, custom item classes are created for item crafting. Examples include ItemGunmetalIngot and ItemPlasticPlate. Instead, we will now use CraftingItem. Here is how a plastic plate looks in the current version:

PlasticPlate = new ItemPlasticPlate();

And it should now look like this:

```
PlasticPlate = new CraftingItem("PlasticPlate", ModernWarfareMod.MODID,
Ores.SyntheticPlastic, 0.5, ModernWarfareMod.BlocksTab);
```

* This example assumes a recovery scrap item of Synthetic Plastic with a yield of 50%

Vests

```
.withPercentDamageBlocked(0.5)
```

Crafting Entries

Without OreDictionary:

```
new CraftingEntry(Items.APPLE, 4)
```

With OreDictionary, the first argument is the default item. This is the item that will be created when dismantled. The second argument is the OreDictionary term.

```
new CraftingEntry(Items.APPLE, "apple", 4)
```

Adding attachment recipes (including bullets)

```
Bullet3006Springfield = new
ItemBullet.Builder().withModId(ModernWarfareMod.MODID)

.withCreativeTab(ModernWarfareMod.AmmoTab).withName("Bullet3006Springfield")
                .withModel(new com.vicmatskiv.mw.models.BulletBig(),
"Bullet.png")
                .withModernRecipe(CraftingGroup.BULLET, new
CraftingEntry(Ores.CopperIngot, 5))
                [...]
```

For Default Recipes:
com/vicmatskiv/crafting/defaultRecipes

Animating Custom Parts

```
setupCustomKeyedPart(ItemAttachment<Weapon> action, String animationFile, String partKey)
```

Sets up an attachment's animations for every animation that it is available for

- **action** is the piece that will be animated
- **animationFile** is the name of the animation file (like how you usually specify it when setting up for setupModernAnimations).
- **partKey** is the name of the part in the animation JSON (in BB)

Example of Setting Up Eject Spent Round
.setupModernEjectSpentRoundAllAnimation(Attachments.MAC10Action, "mac10", "bolt")

This method sets up aimed and non-aimed variants

How to make VMW mobs attack eachother:

Replace “zombieBlistered” with the name of the mob you want it to target

```
.withAiTargetTask(1, e -> new
BetterAINearestAttackableTarget<>((EntityCreature) e, EntityCustomMob.class,
"zombieBlistered", false))
```