Supported Versions

Minimum supported version is **GZDoom 4.5**. Trying to use this library with earlier versions will result in the game not starting since key functionality needed for scaling and rotating are missing. Make sure to set the version number in your ZScript file to this version or higher (e.g. `version "4.5"`).

How to Use

The MAPINFO and ANIMDEFS files define the event handler to use (for rendering the scope) and scope camera texture to use respectively. If you already have one or both present in your own archive, make sure to copy over the contents to the appropriate files. All of the important coding is located in the *scope.zs* file. There is a comment at the top specifying the supported minimum GZDoom version. If you wish to use it standalone, rename it to *zscript.zs* and place it where appropriate with your archive (click here for more info). If you wish to use it with other ZScript and want to keep it in its own file, make sure to `#include` it in your main ZScript file. This library can be used with DECORATE and plenty of functions are present to allow modders to do such, though you'll still need to include the scope file as a ZScript file.

ScopedWeapon

↳ **Weapon**

The base class that all weapons should inherit from that wish to have circular scope functionality. This class will automatically handle tracking the scope camera itself so modders don't have to. The scope camera offsets will correctly offset from the player's current view position taking yaw, pitch, and roll into account. Also includes intelligent support for non-first person cameras so the scope won't continue to be drawn on the screen. Scope rotation, offsetting, scaling, and alpha from the weapon sprite is automatically applied. Only the scope texture is actually rolled with the weapon sprite and has an alpha value. Scope modifying is automatically interpolated based on whether the weapon was that tic. One scope is allowed on the screen at a time and it will always track the main weapon layer (*PSP_WEAPON*). This class will take the player's health and current weapon into account, automatically hiding the scope if they die or switch to a new weapon. This class is multiplayer safe and supports most standard resolutions including scaling with the viewport.

NOTE: Due to Doom's rectangular pixels (all pixels are upscaled vertically by 20%), ~10% of the top and bottom of the scope texture will be cut off in order to preserve this ratio when looking through the scope camera. This should be kept in mind when making scope textures that need an exact fit.

Properties

- *double* **CameraFOV**
  - Sets the FoV to use with the scope. Default is *0*.
- *double* **ScopedWeapon.RenderHeightOffset**
  - The scope is always drawn at the vertical center of the screen. Setting this will offset the scope vertically using the same units as weapon offsets (positive values shift upward). Note that the weapon's default height offset of 32 is taken into account and the scope will generally appear shifted downward. Default is *0*.
- *double* **ScopedWeapon.RenderWidthOffset**
  - The scope is always drawn at the horizontal center of the screen. Setting this will offset the scope horizontally using the same units as weapon offsets (positive values shift to the right). Default is *0*.
- *double* **ScopedWeapon.ScopeScale**
  - Sets the scalar for the scope's size. Default is *1*.
- *string* **ScopedWeapon.ScopeTexture**
  - The name of the texture that should be drawn on top of the scope. Default is no texture.
- *double* **ScopedWeapon.ForwardOffset**
  - How much the scope camera should be offset in the direction the player is looking from the player's current view position (positive values shift forward). This uses regular map units. Default is *0*.
- *double* **ScopedWeapon.SideOffset**
  - How much the scope camera should be offset perpendicular to the direction the player is looking from the player's current view position (positive values shift to the right). This uses regular map units. Default is *0*.

- *double* **ScopedWeapon.UpOffset**
  - How much the scope camera should be offset directly upward from the direction the player is looking from the player's current view position (positive values shift upward). This uses regular map units. Default is *0*.
- *double* **ScopedWeapon.SwaySideMultiplier**
  - Determines how much the horizontal weapon offset should offset the camera perpendicular to the direction the player is looking from the player's current view position. A value of 1 will directly convert the weapon offset to map units (e.g. a weapon offset of 1 means offset 1 map unit to the right). Default is *0*.
- *double* **ScopedWeapon.SwayUpMultiplier**
  - Determines how much the vertical weapon offset should offset the camera directly upward from the direction the player is looking from the player's current view position. A value of 1 will directly convert the weapon offset to map units (e.g. a weapon offset of 1 means offset 1 map unit downward). This will automatically negate the base height offset of 32 that weapons have. Default is *0*.

Functions

- *clearscope Vector2* **GetPrevRenderOffset()** *const*
  - Returns last tic's scope render width/height offsets. Used for interpolating scope movement.
- *clearscope double* **GetPrevFoV()** *const*
  - Returns last tic's scope FoV. Used for interpolating camera zooming.
- *clearscope double* **GetPrevScale()** *const*
  - Returns last tic's scope scale. Used for interpolating scope scaling.
- *clearscope int* **ScopeRenderStyle()** *const*
  - Returns the current player's render style as one of the style constants (see here for possible values). When the player is scoped they are automatically hidden to prevent them from being seen in it. This function will account for that and return the player's true render style.
- *clearscope bool* **ShouldDrawScope()** *const*
  - Returns whether the scope should currently be rendered (accounts for the chase cam).
- *clearscope ScopeCam* **GetCamera()** *const*
  - Returns the pointer to the current scope camera object.
- *clearscope Vector2* **GetWeaponBobOffset()** *const*
  - Returns the current weapon offset from weapon bobbing.
- *clearscope TextureID* **GetScopeTextureID()** *const*
  - Returns the id of the weapon's scope texture.

Action Functions

- *void* **A_Scope(**[*bool doUnscope*]**)**
  - Turns on the scope and optionally toggles it.
    - *doUnscope*
      - If true, allows the function to act as a toggle. Default is *true*.
- *void* **A_Unscope()**
  - Turns off the scope.
- *void* **A_ChangeScopeZoom(**[*double zoomMulti*[*, bool interpolate*]]**)**
  - Changes the scope's FoV using a multiplier of the base FoV.
    - *zoomMulti*
      - The amount to zoom the scope. Default is *1* and resets the scope FoV.
    - *interpolate*
      - If true, interpolates the zooming. Default is *true*.
- *void* **A_ChangeScopeFoV(***[double fov[, bool relative[, bool interpolate]]]***)**
  - Changes the scope's FoV by a flat value.
    - *fov*
      - The amount to zoom by. Positives values zoom in while negative values zoom out. Default is *0* which when used in combination with *relative* resets the scope FoV.
    - *relative*
      - Whether the zoom should offset from the base FoV or set the FoV directly to *fov*. If true, offsets from the base FoV. Default is *true*.
    - *interpolate*
      - If true, interpolates the zooming. Default is *true*.
- *void* **A_ScopeSway(***[double x[, double y[, bool relative]]]***)**
  - Modifies the current scope sway multipliers.
    - *x*
      - The amount to add to the side sway multiplier. Default is *0*.
    - *y*
      - The amount to add to the up sway multiplier. Default is *0*.
    - *relative*
      - Whether to add the values to the sway multipliers or set them directly. If true, adds to the current multipliers. Default is *true*.

- *void* **A_ScopeOffset(***[double f[, double s[, double u[, bool relative]]]]***)**
  - Modifies the current offsets for the actual scope camera position. Uses map units.
    - *f*
      - The amount to offset the camera forward and backward. Positive values shift forward. Default is *0*.
    - *s*
      - The amount to offset the camera left and right. Positive values shift to the right. Default is *0*.
    - *u*
      - The amount to offset the camera upward and downward. Positive values shift upward. Default is *0*.
    - *relative*
      - Whether the value should be added to the current offsets or set them directly. If true, adds to the current offsets. Default is *true*.
- *void* **A_ScopeRenderOffset(***[double x[, double y[, bool relative[, bool interpolate]]]]***)**
  - Modifies the current scope rendering offsets. Uses weapon offset units.
    - *x*
      - The amount to offset left and right. Positive values shift to the right. Default is *0*.
    - *y*
      - The amount to offset upward and downward. Positive values shift upward. Default is *0*.
    - *relative*
      - Whether the value should be added to the current rendering offsets or set them directly. If true, adds to the current offsets. Default is *true*.
    - *interpolate*
      - If true, interpolates the movement. Default is *true*.
- *void* **A_SetScopeTexture(***string name***)**
  - Sets the scope texture to the new one provided by *name*.
    - *name*
      - The name of the texture to use.

- *void* **A_SetScopeScale(***double s[, bool relative[, bool interpolate]]***)**
  - Modifies the current scope scale.
    - *s*
      - The new scale to add.
    - *relative*
      - Whether the value should be added to the current scale or set it directly. If true, adds to the current scale. Default is *true*.
    - *interpolate*
      - If true, interpolate the scaling. Default is *true*.
- *bool* **IsScoped()**
  - Returns true if the weapon currently has a scope. This differs from **ShouldDrawScope()** in that it only checks if the scope camera exists and won't take the chase cam into account.
- *double* **GetScopeZoom()**
  - Returns the current zoom multiplier for the scope based on its default value.
- *double* **GetScopeFoV(***[bool relative]***)**
  - Returns the current zoom level as a flat amount from the default value.
  - *relative*
    - If false, returns the current scope camera FoV instead of the zoom level. Default is *true*.
- *double* **GetScopeForwardOffset()**
  - Returns the current forward scope camera offset in map units.
- *double* **GetScopeSideOffset()**
  - Returns the current side scope camera offset in map units.
- *double* **GetScopeUpOffset()**
  - Returns the current upward scope camera offset in map units.
- *double* **GetScopeWidthOffset()**
  - Returns the current scope rendering width offset in weapon offset units.
- *double* **GetScopeHeightOffset()**
  - Returns the current scope rendering height offset in weapon offset units.
- *double* **GetScopeScale()**
  - Returns the current scale for the scope.
- *string* **GetScopeTexture()**
  - Returns the name of the current scope texture.

- *void* **ClearScopeInterpolation(***[bool rendering[, bool fov[, bool scale]]]***)**
  - Resets the interpolation data for the scope's movement. Can be used to prevent interpolation on that tic.
    - *rendering*
      - If true, clear the scope rendering movement interpolation. Default is *true*.
    - *fov*
      - If true, clear the scope zooming interpolation. Default is *true*.
    - *scale*
      - If true, clear the scope scale interpolation. Default is *true*.