SmoothTurret 2.0 - Script Reference

ST_Player
A basic player script to track if the player is controlling certain turrets or not.

ST_Turret
The main control for turret motion. Can use ballistics, and aim to intercept a moving target. It also has functions to query if the turret is aimed at the target, or if a given target is within the turret's range of motion. This should be placed at the root level of your turret object.

ST_TurretControl
Controls the behavior of ST_Turret, and coordinates its weapons.

ST_TurretSwitcher
A demo script that allows the use of number keys to switch turret behavior and cameras. This is used to swap player control between different turrets.

ST_Player

A simple script to allow a player to control turrets.

Public Variables

turretControl : bool
If true, this player can control turrets. Every turret with *control* set to Player_Mouse or Player_Click and with this script in the *playerScript* field will be controled.

aimObject : GameObject
The object that a controlled turret will aim at. Uses raycast to find a location the mouse is pointing at.

fireObject : GameObject
If ST_Turret *control* is Player_Click, this is the object placed when clicking. The turret will then fire at that object.

emptyAimRange : float
The distance to place *aimObject* if aiming at a spot with no collider.

fixedAimObjectRange : bool
aimObject always appears at Empty Aim Range.

ST_Turret : MF_AbstractPlatform

Describes the properties of a turret. Requires ST_TurretControl.

Inherited Variables

target, weaponMount, aimObjectActive, aimObject, emptyAimRange, fixedAimObjectRange, useIntercept, useVelocityZero, shotSpeed, exitLoc, targetAimLocation, targetRange, lastTargetPosition, targetRigidbody, velocity, playerControl

Inherited Methods

Override **AimLocation** : Vector3 ()
Adjusts aim location based on intercept and ballistic settings.

Override **TargetWithinLimits** : bool (*targ* : Transform)
Returns true if *target* is within the turning limits of the turret.

Override **AimCheck** : bool (*targetSize* : float, *aimTolerance* : float)
Returns true if the turret is aimed properly to hit the target. This takes into account the intercept and ballistics.

Public Variables

useGravity : bool
Aim as if the shots will be affected by gravity. Assumes zero drag.

defaultArc : ArcType enum { Low, High }
When firing ballistic shots affected by gravity, most firing solutions will have both a high arc and a low arc. SmoothTurret will pick the specified arc. High arc requires more computation cycles to achieve reasonable accuracy.

ballisticIterations : int
Using intercept and ballistics at the same time is an iterative function. This will specify how many time to run the process for the lower arc trajectory. Higher numbers result in greater accuracy, at the expense of processor usage. Longer flight times generally require more iterations to achieve acceptable accuracy. For the lower arc, 1 is likely to be acceptable for most applications.

highArcBallisticIterations : int
Using intercept and ballistics at the same time is an iterative function. This will specify how many time to run the process for the higher arc trajectory. Higher numbers result in greater accuracy, at the expense of processor usage. Longer flight times generally require more iterations to achieve acceptable accuracy. Since the high arc flight times tend to be much larger than the low arc, the default is set to 5.

rotationRateMax : float (1 to 720)
elevationRateMax : float (1 to 720)
The maximum rate in degrees per second that each part may turn. Note, that often times your turret may not reach higher maximum turn rates. This is due to two main factors. First, the turret will also need time to slow down at some point before it reaches its destination, effectively limiting how fast it can turn. Second, there will typically only be 180 degrees max to rotate to its goal. Between the time needed to accelerate and slow down, there may not be enough degrees to hit the max turn rate before reaching its goal.

These will also have an effect on the pitch/rate of the *rotator* and *elevator* motion sounds.

rotationAccel : float (1 to 720)
elevationAccel : float (1 to 720)
The acceleration rate of the *rotator* and *elevator* parts of the turret. This is how fast that part speeds up and slows down its turning. Acceleration rates above 500 will begin to cause some slight jitter, with it getting more noticeable around 700. High acceleration rates also begin to suffer diminishing returns. To get true high speed turning, consider using constant turn rate, as this will allow very fast and precise turning, as well as eliminating high speed jitter.

decelerateMult : float
This allows you to cause deceleration to be a different rate than accelerating. It affects both rotation and elevation. Typically, this value will be used to simulate braking being easier than accelerating.

constantTurnRate : bool
This allows you to forgo smooth natural movement, and have the turret simply use constant turn rates. This setting uses less computation, and is also better for turrets using very fast turn rates. Accuracy is also better than using smooth turning.

If this option is used, the turret will turn at the rate supplied in *rotationAccel* and *elevationAccel* but will still be limited by *rotationRateMax* and *elevationRateMax*.

limitLeft : float (0 to -180)
limitRight : float (0 to 180)
limitUp : float (0 to 90)
limitDown : float (0 to -90)
Each of these allow you to place traversal limits on the rotator and elevator parts. Movement tracking left and down use negative numbers. Tracking to the right and up use positive numbers.

restAngle : Vector2
You may define an angle for the turret to point when it has no target.
x is the angle of elevation (-90 to 90).
y is the angle of rotation (-180 to 180).
0, 0 points straight ahead.

restDelay : float
The time in seconds a turret will remain at its current position without a target before turning to its *restAngle*.

aimError : float
aimErrorInterval : float
Optionally, you can give the turret some inaccuracy in rotation. *aimError* is the inaccuracy in degrees, and *aimErrorInterval* is how often a new random ‘wrong’ position is calculated. These values typically look best when under 1, with the interval looking better higher when turn rates are slower.

turningWeapInaccuracy : float
Adds to weapon inaccuracy if the turret is turning, for every degree per second, from both the elevator and rotator parts. ST_TurretControl will send this value to every weapon in the turret. But will be bound by each individual weapon's *inaccuracy*, and *maxInaccuracy* found in ST_TurretControl.

rotator : Transform
The object that rotates around the y axis.

elevator : Transform

The object that rotates around the x axis.

rotatorSound : AudioObject
elevatorSound : AudioObject
These control the audio to be used for each part's movement sound. This sound will play at a varying pitch/rate based on the current rotation/elevation speed as compared to its respective maximum.

For best results, this sound should be a constant looping whir or hum. Steady looping rhythmic elements work well too, as their rate will also vary with the turning speed.

Hidden Variables

systAimError : Vector3
The current location of any error introduced with *aimError*. This will change every *aimErrorInterval* seconds.

averageRotRateEst : float
averageEleRateEst : float
Used to smooth the current rotation and elevation reading because of spikes from inconsistent frame rates.

totalTurnWeapInaccuracy : float
The amount of inaccuracy ST_TurretControl will send to weapons resulting from *turningWeapInaccuracy* * (*averageRotRateEst* + *averageEleRateEst*)

curArc : ArcType enum { Low, High }
The current ballistic arc being used. This can be modified by ST_TurretControl if the default arc LoS is blocked, and losMayChangeArc is used,

Public Class AudioObject

audioSource : AudioSource
The audio source used for audio.

pitchMax : float
The maximum pitch the audio can be played at. This is reached when turning at full speed.

pitchMin : float
The minimum pitch the audio can be played at. This is the pitch that is approached as the turning motion slows.

ST_TurretControl : MF_AbstractPlatformControl

Controls the behavior of ST_Turret, and coordinates its weapons. This may be used with MF_B_Turret or ST_Turret.

Inherited Variables

targetingScript, weapons, curWeapon

Inherited Classes

WeaponData

Public Variables

target : GameObject
Current target of the turret. This will either be supplied by the targeting script, or be assigned directly. However, if the turret is choosing its own targets, this may immediately be overwritten.

controller : ControlType enum { AI_AutoTarget, None, Player_Mouse, Player_Click }
AI_AutoTarget: AI will use a scanner to pick targets.

None: Turret is deactivated.

Player_Mouse: Turret will aim at the defined player's *aimObject*, and will fire when the player presses the left mouse button. That player must also have *turretControl* set to true. If false, the turret will not fire or move.

Player_Click: Turret will aim and fire at a clicked location. If *fullBurst* is true, it will fire a full burst according to the weapon. Otherwise, a single shot is fired.

playerScript : ST_PlayerScript
Used when *controller* is set to Player_Mouse or Player_Click. This will give the turret that player's *aimObject* as its target.

fixedConvergeRange : float
When using multiple weapons, you may designate a range that weapon fire will converge at using *fixedConvergeRange*. This will make the weapons angle slightly during Start() at runtime.

dynamicConverge : bool
If true, weapons will ignore *fixedConvergeRange*, and continuously angle to produce convergence at the range of the current target.

convergeSlewRate : float
Determines how fast the individual weapons will angle in degrees per second during *dynamicConverge*.

minConvergeRange : float
You may also designate a minimum convergence range, so as to limit the amount that the individual weapons can move.

alternatingFire : bool

When using multiple weapons, you can choose to have them all fire at once, or alternate shots between them. If alternating, the delay before the next weapons fires is the *cycleTime* of the weapon at index 0 divided by the number of weapons. This will produce odd results if you have weapons with different rates of fire.

fullBurst : bool
If true, when the weapon fires, a full burst will be unleashed according to the settings of MF_AbstractWeapon. If 0 or 1 is selected in *burstAmount*, it will still function as a single shot.

checkLOS : bool
Select if the turret should use line of sight to determine whether or not to fire. If using gravity, LoS will use a ballistic arc approximation. (Raycast from gun exit, to apex of the arc, to the target.)

alwaysDirectLos : bool
If true, LoS checks will always be directly from the gun exit to the target, and will not approximate a ballistic arc.

losMayChangeArc : bool
If true, when using gravity, if the LoS is blocked along the default arc, the turret will attempt to try the other arc. This will reset back to the default after every shot or change of target.

losCheckInterval : float
If performance becomes an issue, you can define an interval to wait between checks, although this can result in the weapon being less responsive to rapidly changing line of sight conditions. 0 = check before every shot.

maxInaccuracy : float
If the turret's *turningWeapInaccuracy* is > 0, inaccuracy will be sent to the weapons. This will limit the amount of inaccuracy a weapon can have.

checkTargetSize : bool
This script tries to determine the target size when choosing whether or not to fire. This way, it will fire even if not aimed directly at the target center. It approximates the size of the target using its largest dimension of any root collider component. If no collider is found, it will look for a collider in the root child index 0. If still none found, it will use *targetSizeDefault*.

targetSizeDefault : float
If no target size can be found, you can set a default size for the the weapons to fire at.

ST_TurretSwitcher

A simple script to allow a player to use number keys to switch between turrets.

Public Variables

ActivatedBehavior : ControlType enum { AI_AutoTarget, None, Player_Mouse, Player_Click }
How the turret acts when its number key is pressed.

DeactivatedBehavior : ControlType
How the turret acts when other number keys are pressed.

switchCamera : bool
If true, will witch to the camera assigned to a turret when activated.

turrets : TurretData[]
An array of turrets to switch between. This should not be more than 9 turrets.

selectedTurret : int
The index of the currently selected turret.

playerScript : ST_Player
If blank: Will assume the player script is on the same object. May also be left blank if none of the behaviours use a player script.

Public Class TurretData

turretScript : ST_TurretControl
The turret assigned to this index.

turretCamera : GameObject
The camera to use when this index is activated.