Zed Programming Documentation

Creators:

Teddy Walsh (Teddywalsh72@gmail.com) (I'm the one writing this right now)

Sijal Jardat

Intro:

Zed has a lot of code! As of writing this, zed.cs has 1,224 lines of code. This will try to go over his major systems in as organized of a manor as I can.

Naming Convention

Variables that refer to Zed's attacks use an acronym for each attack (note: there is a name key in a comment at the top of Zed.cs to refer to). Also, most of the attacks have a phase 1 version and a phase 2 version, so "P1" and "P2" are used respectively.

- "CSP1" = "Charged Slash Phase 1
- "SJP2" = "Super Jump Phase 2"

These names are also used in the zedAttacks enum (recorded in the currentAttack variable)

Also, every debug line in Zed.cs has the word "zed" in it, so while testing, its good to put "zed" into the search bar of the console to filter out any other debug lines that you don't need to care about.

Attack Hitboxes

Normally, an enemy (like the corrupted individual) can have 1 hitbox prefab that it spawns and destroys every time it attacks, but Zed has multiple different attacks that he does and many of the attacks have multiple hitboxes.

Each hitbox has a corresponding [SerializeField] GameObject variable that points to a different hitbox prefab. These variables should already be assigned in Zed's prefab. Each of these hitbox prefabs is a game object with the hitbox script, which does all the logic of hit detection. All Zed.cs has to do is create them and destroy them at the right time.

Zed's attack animations will call EnableHitbox() and DisableHitbox() at certain frames. EnableHitbox() will instnatiate a new instance of the correct hitbox with Zed's transforms and

save it in the variable CurrentAttackHitbox. The exceptions to this are for Super Jump Phase 2, which also creates shockwaves around where he lands, and Shooting Sword Phase 1, which creates the hitbox at the projectile's transform rather than his own, and makes the sprite visible as well. Lastly, it flips the orientation of the hitbox if Zed should be facing left. It also records hitboxesThisAttack incase there is an attack with multiple different hitboxes it needs to spawn (like multiple swings). This feature isn't used currently but its there if you need it.

DisableHitbox() will destroy currentAttackHitbox and in the case of shooting sword phase 1, will also make the projectile sprite invisible again. Note: DisableHitbox() is also called when Zed is parried and/or stunned.

If EnableHitbox() is called twice before DisableHitbox() is called, then the first hitbox will no longer be recorded with CurrentAttackHitbox() and will never be deleted. DisableHitbox() is called before every attack ends and every time an attack is interrupted (like if Zed is stunned).

Attack Movement

Many of Zed's attacks require him to move as part of the animation. All of the movement is handled by the code and the animations themselves should have Zed centered in the frame the whole time.

Each attack that requires Zed to move has a few variables that should be assigned in Zed's prefab. Note that "X" refers to the name of the attack in all of the following variables:

- "XMovementDifRight" This variable is a vector 3 that holds how far from his starting position should Zed move by the time he ends this attack movement.
- "XMovementDifLeft" is assigned in Start() and is the value entered for "XMovementDifRight" except the X value is negative.
- "XMovementTime" this is the time (in seconds) that it should take Zed in total to move from the beginning to the end. This should be assigned in the prefab and calculated based on the ammount of frames he should be moving in the animation / the framerate of the animation, so if Zed is animated at 12 FPS and he starts moving on frame 8 and finishes moving on frame 20, then XMovementTime should = ((20 - 8) / 12) = 1. This variable is only used to calculate XMovementSpeed, which is whats actually used in the code to move Zed.
- "XMovementSpeed" the distance Zed should move per frame. It is assigned in Start() based on the magnitude of XMovementDifRight and XMovementTime.

AttackMove() is called by animation frames when Zed needs to move. It marks isMoving to true, which has effects in FixedUpdate() that will be covered shortly. It also records how many times Zed has moved so far in this attack in the event that the attack has multiple moves like Super Jump where Zed goes up into the air first, then slams down second, or Shooting Sword P1 where he fires the sword out first, and it returns second. After that, based on what attack it is and which move its on, it calculates targetPosition with either XMovementDifRight or XMovementDifLeft depending on which way Zed is facing, and assigns movementSpeed based on that as well.

In FixedUpdate(), while currentState == zedStates.attacking and while isMoving is true, Zed will move towards his target position based on his movementSpeed. Note: movementSpeed is already calculated for distance/frame and this is in FixedUpdate(), so theres no need for DeltaTime. Lastly, if Zed is within stopDistToTarget distance of his target, his movement is stopped.

Note: The two exceptions to these patterns are

- SuperJumpFall (both P1 and P2), which locks onto the players position rather than using a set distance from Zed's starting position, so their movementSpeed are calculated each time AttackMove() gets called for it, and they don't have an XMovementDifRight variable.
- ShootingSword P1, which moves the SSProjectile instead of Zed himself and uses projectileOverride.

Zed's AI

Most of Zed's decision making is done in FixedUpdate (this is one of the things that could likely be reworked in the future) and is heavily based on currentState. The logic behind his AI is determined by the flowchart on this page.

If Zed is stunned, has not started the fight yet, is dead, or otherwise, then he should not do anything. If he is attacking, then possibly move him if isMoving is true. If he is idle, then he decides what his next attack should be. First, roll a random number between 1 and 100 (inclusive) to simulate a percent chance (which is what the game design flowchart outlined). Then, check if Zed is in phase1 or phase2, and also check if the player is nearby or far away (based on the closeRangeDist variable, and playerDist, which is updated every frame).

> For example, in phase 1, while the player is close, there is a 40% chance to do a ChargedSlash, a 50% chance to do RapidStrikes and a 10% chance to do a superJump, so based on the number, 1-40 = ChargedSlash, 41-90 = RapidStrikes and 91-100 = SuperJump.

Each attack has a function that is called to start it in FixedUpdate once Zed decides he will do that attack. Each function first sets the corresponding trigger to Zed's animation controller (the variable "Animator"). Then, it will set the currentAttack to the correct attack, often parsing between different versions of the attack for phase 1 and phase 2. Then, it will call StartAttack() which has some cleanup code. The rest of the attack logic is handeled by the animations and the animationEvents they call.

Note: Some attacks are split up into multiple steps becasue Zed has logic about continuing them or not, such as for RapidStrikes, Zed will do the first attack, then if the player is still in front of him, he will do the second, otherwise he will go back to idle, so these can't all be 1 animation.

An attack animation will call EnableHitbox() and DisableHitbox() based on its own timing, and if it needs to move, it will call AttackMove() also based on its own timing. All of these have been covered in detail. The last thing a every attack animation does, is on the very last frame, it

calls EndAttackAnimation(). This covers logic like playing change-back animations and doing follow-up attacks (like RapidStrikesSecond(). It also resets currentState back to idle to engage Zed’s AI again and checks if he should move to phase 2.

Ultimate

I have no idea how this works lmao. Ask Sijal :/