Spirit Data: How it Works

Spirit data has now been implemented in the game, and tons of things were added and changed along the way. I'll give a general overview on how everything works and explain some things that were reworked to make this work.

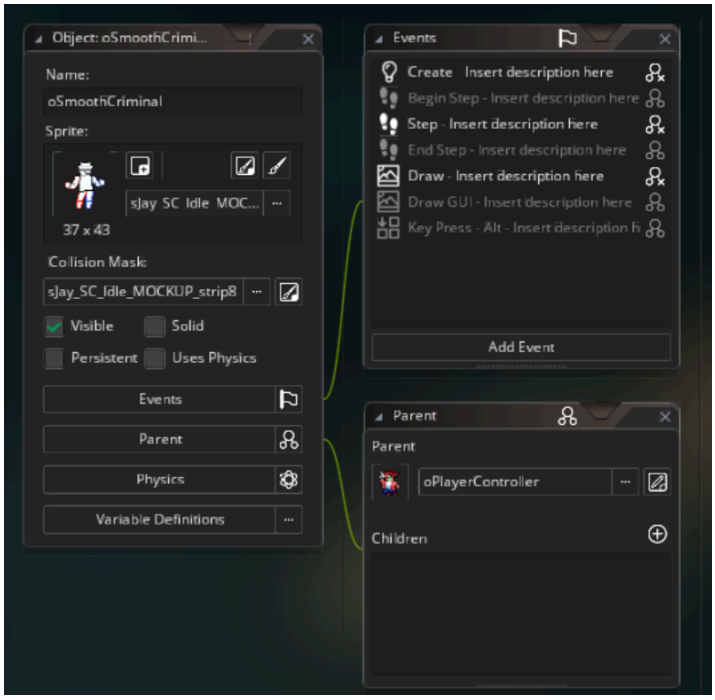
Index:

Jay now has a spirit that fights alongside him, called Smooth Criminal (abbreviated to SC). There are a ton of rules that he follows to keep him balanced, complex, and fun.

- Jay switches between two different states: Spirit OFF and Spirit ON. These two states switch between two completely different movesets.
    - In Spirit OFF, Jay fights by himself and SC shows up for only a few moves. Jay is a bit slow and his attacks are weak here.
    - In Spirit ON, SC stands by Jay's side and fights in his stead. SC follows Jay's movement and he has his own attacks. Jay can double jump in this state too.
- SC has his own health bar. He will take damage whenever Jay gets hurt in Spirit ON. He recovers health over time whenever he's away in Spirit OFF. If he runs out of health, Jay enters a Spirit BREAK state where he is locked to his Spirit OFF moveset and he is unable to summon SC until SC's health fully regenerates again.
- To summon SC and switch to Sprint ON, use Neutral Special. The enhancer will make SC do a lunge punch that goes through the opponent, allowing him to stay a certain distance away from Jay for sandwich pressure.
- Jay has two supers, one in Spirit OFF and one in Spirit ON.
    - In Spirit OFF, Jay enters a special state where he and SC can attack at the same time, essentially combining Spirit OFF and Spirit ON. This allows Jay to maximize his offensive pressure and combos. SC doesn't take damage in this state and this state only lasts for a certain amount of time. Jay cannot activate this state when he is in Spirit BREAK.
    - In Spirit ON, SC stops time. This move has a long startup, so it is hard to set it up or combo into it. However, if it connects, the opponent is frozen in place and Jay and SC can pummel them. The time stop rapidly drains Jay's meter, and the time stop ends when Jay runs out of meter.

The main goal of this overall design is to make Jay a puppet character where he needs to fight alongside SC effectively to keep the upper hand in a fight.

To begin, Smooth Criminal is his own character that inherits from oPlayerController. This is because he has his own data and attacks separate from Jay.



Create, and Step are different from the rest of the characters. Draw was edited to remove shader data for now because Smooth Criminal doesn't have any palette data at the moment.

Create simply makes some new variables and creates a copy of its moveset, similar to Jay. This is for certain moves Smooth Criminal has that will be explained later.

```
// Spirit exclusive data
host = noone;
hostObject = noone;
nextToPlayer = true;
vulnerable = false;
hurtboxSet = false;
attack = 0; // Matches Jay's attack

// Variables for when spirit is summoned in Spirit OFF
inSpiritOff = false;
startingMove = 0;

selectedCharacter = -1;
for (var i = 0; i < global.numberOfCharacters; i++;)
{
    if (global.characterData[i].Name == "SmoothCriminal")
    {
        selectedCharacter = StructCopy(global.characterData[i]) // This is to make sure that each player has a seperate copy of their move data
        break;
    }
}

event_inherited();
```

All of the host variables refer to Jay, since he is Smooth Criminal's host. They will constantly transfer data back and forth with each other. The variable host refers to Jay the character with the move data and hostObject refers to Jay the game object with movement and states.

Step however, has many differences from oPlayerController/Step. So much so that it doesn't even use event_inherited() because that would complicate and break things. I'll explain the changes as I go.

Jay also has some new data in oCharacterController/Create to link him to Smooth Criminal (SC).

```
334 spiritState = false; // false = Spirit OFF, false = Spirit ON
335 spirit = noone;
336 spiritObject = noone;
337 spiritSummoned = false;
338 spiritBroken = false; // Host cannot summon spirit when the spirit loses all its health
339 spiritMaxHealth = 0;
340 spiritCurrentHealth = 0;
341 spiritRegenSpeed = 0;
342 spiritKORegenSpeed = 0;
343 if (selectedCharacter.UniqueData.SpiritData == 1)
344 {
345     for (var i = 0; i < array_length(global.characterData); i++)
346     {
347         if (selectedCharacter.UniqueData.Spirit == global.characterData[i].Name)
348         {
349             spirit = global.characterData[i];
350             spiritMaxHealth = spirit.MaxHP;
351             spiritCurrentHealth = spiritMaxHealth;
352             spiritRegenSpeed = spirit.RegenSpeed;
353             spiritKORegenSpeed = spirit.KORegenSpeed;
354             break;
355         }
356     }
357 }
358 // If host is trying to summon/unsummon spirit but gets interrupted, spirit gets summoned after hitstop
359 pendingToggle = false;
```

The variable spirit refers to SC the character with the move data and spiritObject refers to SC the game object with movement and states.

So to begin with on how SC works, there are a number of functions that were omitted from SC's Step to prevent duplication of events.

- SC does not have his own attack button. Instead, all of his attacks are executed when Jay executes moves. Since Jay has to pose when SC does attacks in Spirit ON, those poses are moves without hitboxes and SC attacks in tandem with them. This is triggered in PressAttackButton.

```
function PressAttackButton(attack)
{
    // Input Normal attacks first
    switch attack
    {
        case 1:
            if (prevState == eState.STANDING_LIGHT_ATTACK_2)
            {
                state = eState.STANDING_LIGHT_ATTACK_2;
                SetSpiritMoveData(false, selectedCharacter.StandLight2, attack);
            }
            else if (prevState == eState.STANDING_LIGHT_ATTACK_3)
            {
                state = eState.STANDING_LIGHT_ATTACK_3;
                SetSpiritMoveData(false, selectedCharacter.StandLight3, attack);
            }
            else if (state == eState.JUMPING || state == eState.RUSH_CANCEL_UP)
            {
                state = eState.JUMPING_LIGHT_ATTACK;
                SetSpiritMoveData(false, selectedCharacter.JumpingLight, attack);
            }
            else if (verticalMoveDir = -1)
            {
                state = eState.CROUCHING_LIGHT_ATTACK;
                SetSpiritMoveData(false, selectedCharacter.CrouchingLight, attack);
```

```
// Update the spirit's move whenever the host performs an attack
function SetSpiritMoveData(enhancement, move, attack)
{
    if (spirit != noone && spiritSummoned && move.SpiritData.PerformAttack)
    {
        spiritObject.state = state;
        if (!enhancement)
        {
            spiritObject.attack = attack;
        }
        spiritObject.animTimer = 0;
    }
    if (move.SpiritData.ReturnToPlayer && spiritState)
    {
        spiritObject.nextToPlayer = true;
    }
}
```

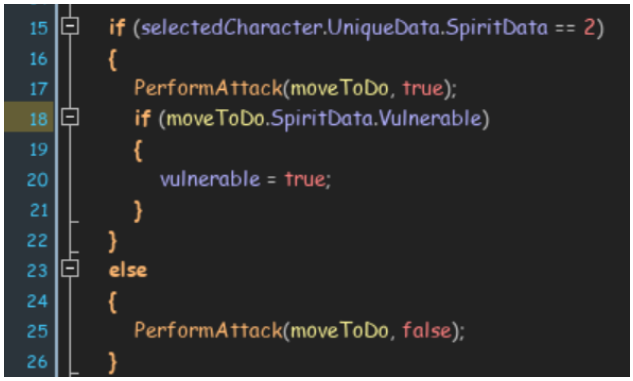
This ensures that Jay and SC are in sync and it prevents potential desync infinites.

- I'm unsure if SC should be able to grab his opponents, so for now I made it so he can't by not matching Jay's grab state with SC's grab state.
- SC is unable to perform Rush Cancels. Instead, when Jay performs them, SC enters the Rush Cancel states with him. This is to prevent the player from spending 100 meter to perform Rush Cancels when SC is activated.

If you look through the code, you'll notice that SC still has his own movement. The movement still mirrors that of Jay's, but this also allows for potential movement desyncs (for example, if Jay and SC sandwich the opponent, Jay could backdash while SC performs a running jump). I decided to include this for experimentation and potential advanced tech.

For the attacks themselves, there is a lot to go over for how attacks work with SC. Since I edited and added a number of scripts.
In the Ground/Crouching/JumpingAttackScripts, I added a ton of code for summoning/deactivating spirits.

- Firstly, spirits have a Vulnerable variable where if they perform a move that has this variable set to true, it would die instantly. This is checked when running PerformAttack.



```
if (selectedCharacter.UniqueData.SpiritData == 2)
{
    PerformAttack(moveToDo, true);
    if (moveToDo.SpiritData.Vulnerable)
    {
        vulnerable = true;
    }
}
else
{
    PerformAttack(moveToDo, false);
}
```

- If you perform a move that activates/deactivates Spirit ON, the state will switch when the move ends. However, if Jay is interrupted before the move ends, the state is still supposed to switch. For this reason, pendingToggle is used to make this work.

```
// The purpose of this is to actiate/deactivate spirit if a toggle move is performed but it's interrupted before it ends
if (selectedCharacter.UniqueData.SpiritData == 1 && moveToDo.SpiritData.ToggleState && !spiritBroken)
{
    pendingToggle = true;
}
```

- Next, it checks if Jay's move has the spirit attack in Spirit OFF, which leads to its own script.

```
// If this move temporarily summons the spirit to attack in Spirit OFF
if (selectedCharacter.UniqueData.SpiritData == 1 && !spiritState && moveToDo.SpiritData.PerformInSpiritOff && !spiritBroken)
{
    SummonInSpiritOff(moveToDo);
}
```

```
function SummonInSpiritOff(moveToDo)
{
    SummonSpirit();
    spiritObject.x += moveToDo.SpiritData.StartXOffset * image_xscale;
    spiritObject.y += moveToDo.SpiritData.StartYOffset;
    if (!moveToDo.SpiritData.SummonSpirit)
    {
        spiritState = false;
    }
    else
    {
        if (selectedCharacter.UniqueData.LinkMovesetsWithSpirits)
        {
            currentMovesetID = selectedCharacter.UniqueData.SpiritOnMoveset;
            OverwriteMoveset();
        }
    }

    with (spiritObject)
    {
        OverwriteSpiritMoveset(true);
    }
    spiritObject.inSpiritOff = true;
    spiritObject.state = state;
    spiritObject.startingMove = state;
    pendingToggle = false;
}
```

SummonInSpiritOff begins by running another script called SummonSpirit.

```
function SummonSpirit()
{
    if (!spiritSummoned)
    {
        if (spirit.Name == "SmoothCriminal")
        {
            spiritObject = instance_create_layer(x + (10 * image_xscale), y, "Instances", oSmoothCriminal);
        }
        spiritObject.host = selectedCharacter;
        spiritObject.hostObject = self;
        spiritObject.playerID = playerID;
        spiritObject.opponent = opponent;
        spiritSummoned = true;

        spiritObject.image_xscale = image_xscale;
        var spiritFire = instance_create_layer(spiritObject.x, spiritObject.y, "Instances", oSpiritFire);
        spiritFire.depth = depth + 1;
    }
    spiritState = true;
    pendingToggle = false;
    if (selectedCharacter.UniqueData.DoubleJump)
    {
        canDoubleJump = true;
    }
}
```

SummonSpirit creates an instance of the SC object and sets all of the necessary data so Jay can control him.
SummonInSpiritOff is supposed to keep Jay in Spirit OFF while still summoning SC, so it reverts some data. However, if the move is supposed to summon SC proper, then it switches the moveset too and it runs the following block of code in SC's Step.

```
    // CHanges the moveset to its Spirit ON version
    if (inSpiritOff && state != startingMove && state != eState.HITSTOP && hostObject.spiritState)
    {
        OverwriteSpiritMoveset(false);
        inSpiritOff = false;
    }
```

Once SC's move ends in Spirit OFF and it enters the Idle, Crouch, Walk, Run (both), or Jump/Jumpsquat states, it runs a script that destroys it called DeactivateSpirit.

```
function DeactivateSpirit(executedBySpirit)
{
    if (!executedBySpirit)
    {
        if (spiritSummoned)
        {
            instance_create_layer(spiritObject.x, spiritObject.y, "Instances", oSpiritFire);
            instance_destroy(spiritObject.hurtbox);
            instance_destroy(spiritObject);
            spiritObject = noone;
            spiritSummoned = false;
        }
        spiritState = false;
        pendingToggle = false;
        if ((selectedCharacter.JumpType & 1) != 1)
        {
            canDoubleJump = false;
        }
    }
    else
    {
        hostObject.spiritObject = noone;
        hostObject.spiritSummoned = false;
        hostObject.spiritState = false;
        if ((host.JumpType & 1) != 1)
        {
            hostObject.canDoubleJump = false;
        }
        if (host.UniqueData.LinkMovesetsWithSpirits)
        {
            hostObject.currentMovesetID = host.UniqueData.SpiritOffMoveset;
            with(hostObject)
            {
                OverwriteMoveset();
            }
        }
        with(hostObject.hurtbox)
        {
            spiritOwner = noone;
        }
        instance_create_layer(x, y, "Instances", oSpiritFire);
        instance_destroy(hurtbox);
        instance_destroy();
    }
}
```

It runs two different versions: one where SC himself is activating it, and one where something else activates it. Either way, it destroys the SC object and reverts Jay back to Spirit OFF.

- The next if statement is moreso a failsafe than anything else. It automatically deactivates SC if he's doing a move in Spirit OFF and Jay cancels into another attack that doesn't summon SC.

```
// If the current move doesn't have the spirit perform a move in Spirit OFF and it's around, destroy it
if (selectedCharacter.UniqueData.SpiritData == 1 && !spiritState && spiritObject != noone && !moveToDo.SpiritData.PerformInSpiritOff)
{
    DeactivateSpirit(false);
}
```

- Now we enter the if statement that runs when the move finishes. First, the moveset switching code was modified a little bit to factor in spirits if the character has one. In this case, Jay does.

```
// If this move updates the moveset, switch the moveset
if (selectedCharacter.UniqueData.AdditionalMovesets > 0) // If this character has multiple movesets...
{
    if (moveToDo.SwitchMoveset)
    {
        if (selectedCharacter.UniqueData.LinkMovesetsWithSpirits && !spiritBroken)
        {
            currentMovesetID = moveToDo.SwitchToMoveset;
            OverwriteMoveset();
        }
        else if (!selectedCharacter.UniqueData.LinkMovesetsWithSpirits)
        {
            currentMovesetID = moveToDo.SwitchToMoveset;
            OverwriteMoveset();
        }
    }
}
```

- Finally, it runs the code that activates/deactivates SC when the move is supposed to do so completely.

```
73      // If this move switched Spirit state
74      if (selectedCharacter.UniqueData.SpiritData == 1 && moveToDo.SpiritData.ToggleState && !spiritBroken)
75      {
76          if (!spiritState)
77          {
78              SummonSpirit();
79          }
80          else
81          {
82              DeactivateSpirit(false);
83          }
84      }
```

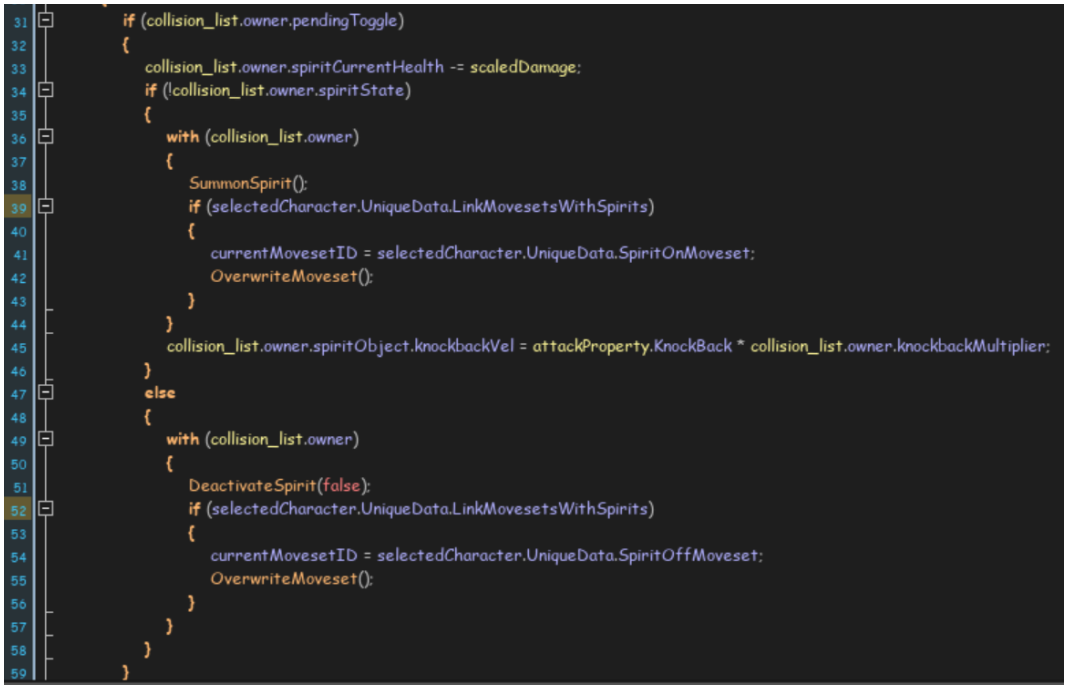
When SC performs an attack, slightly modified versions of PerformAttack and oHitbox/Step2 play out.

- I set up SC's hurtboxes and hitboxes so they are actually an extension of Jay's. This makes it easy to program things like combos between them and having them both get hit at the same time. In order to set this, I set the owner of the hitboxes and hurtboxes to Jay and then I created a variable in the hitbox and hurtbox objects that get the spirit so they can get the spirit's position.
- oHitbox/Step2 doesn't have much that's different about it, but I made a small change where if SC attacks the opponent, it turns the opponent towards him instead. The reason for this is for knockback to go in the correct direction. I also modified ProcessHit for this purpose.
- In ProcessHit, I changed how knockback works so the direction the player flies is based on their own orientation rather than the attacking player's.

```
101     else if (attackProperty.Launches)
102     {
103         collision_list.owner.vsp = attackProperty.LaunchKnockbackVertical * collision_list.owner.knockbackMultiplier;
104         // Horizontal direction of destructible objects are based on the player hitting them
105         if (collision_list.owner.isDestructibleObject)
106         {
107             collision_list.owner.hsp = attackProperty.LaunchKnockbackHorizontal * collision_list.owner.knockbackMultiplier * owner.image_xscale;
108         }
109         else
110         {
111             collision_list.owner.hsp = attackProperty.LaunchKnockbackHorizontal * collision_list.owner.knockbackMultiplier * -collision_list.owner.image_xscale;
112         }
113         collision_list.owner.grounded = false;
114
115         if (collision_list.owner.spiritObject != noone && collision_list.owner.spiritState)
116         {
117             collision_list.owner.spiritObject.vsp =  attackProperty.LaunchKnockbackVertical * collision_list.owner.knockbackMultiplier;
118             collision_list.owner.spiritObject.hsp = attackProperty.LaunchKnockbackHorizontal * collision_list.owner.knockbackMultiplier * -collision_list.owner.spiritObject.image_xscale
119             collision_list.owner.spiritObject.grounded = false;
120         }
```

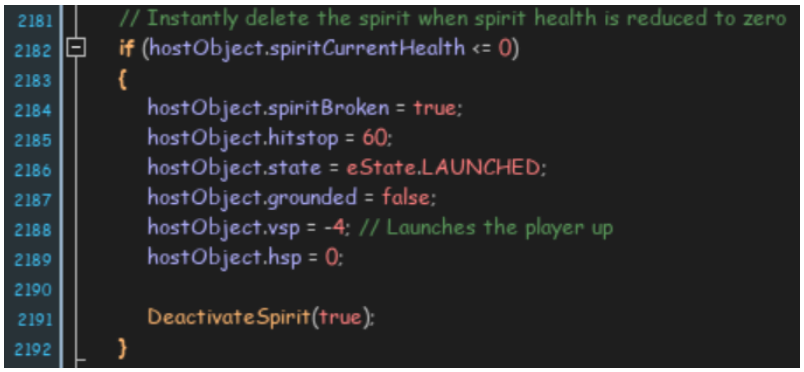
This is also where pendingToggle is put to use. If Jay gets hit while trying to

toggle his spirit state, it will toggle immediately and SC will take damage.



```
if (collision_list.owner.pendingToggle)
{
    collision_list.owner.spiritCurrentHealth -= scaledDamage;
    if (!collision_list.owner.spiritState)
    {
        with (collision_list.owner)
        {
            SummonSpirit();
            if (selectedCharacter.UniqueData.LinkMovesetsWithSpirits)
            {
                currentMovesetID = selectedCharacter.UniqueData.SpiritOnMoveset;
                OverwriteMoveset();
            }
        }
        collision_list.owner.spiritObject.knockbackVel = attackProperty.KnockBack * collision_list.owner.knockbackMultiplier;
    }
    else
    {
        with (collision_list.owner)
        {
            DeactivateSpirit(false);
            if (selectedCharacter.UniqueData.LinkMovesetsWithSpirits)
            {
                currentMovesetID = selectedCharacter.UniqueData.SpiritOffMoveset;
                OverwriteMoveset();
            }
        }
    }
}
```

If SC runs out of health, he is forcefully deactivated and Jay enters a Spirit Broken state, where he cannot summon SC until he recovers. Activating this is handled in Sc's Step.



```
// Instantly delete the spirit when spirit health is reduced to zero
if (hostObject.spiritCurrentHealth <= 0)
{
    hostObject.spiritBroken = true;
    hostObject.hitstop = 60;
    hostObject.state = eState.LAUNCHED;
    hostObject.grounded = false;
    hostObject.vsp = -4; // Launches the player up
    hostObject.hsp = 0;

    DeactivateSpirit(true);
}
```

Finally, because SC has his own health, he has his own health bar in the HUD. This health bar will only show up if the player is playing as Jay.