Beginning Notes:

If you have any specific questions not listed here or suggestions/comments for the scripting guide, you can DM me on reddit at /u/MrEpicIsHere777. This guide was created with notes and help from many people of the modding community, it is a collective effort that I assembled into a document.

Before you read this, I highly recommend reading this guide to getting started with a DDLC mod. This document you are reading now is more intended to be a "cheat sheet" while you script.
Read it here:
https://www.reddit.com/r/DDLCMods/comments/89tf1p/getting_started_with_your_ddlc_mod/

I also highly recommend using the Ren'Py doc to guide you along your scripting as well.
Read it here:
https://www.renpy.org/doc/html/index.html

If I write something along the lines of "<character>" anywhere I put them, do not include <> in your script. It's for guide purposes only.

With that said, enjoy the doc.

~ MrEpic

Character Posing + Position:

Body Pose (add b for casual outfit)	Sayori	Natsuki	Yuri	Monika
1	Default	Default	Default	Default
2	Right Arm Raised	Right Hand on Hip	Right Hand Up	Hand on Hip
3	Left Arm Raised	Left Hand on Hip	Both Hands Up	Pointing
4	Both Arms Raised	Both Hands on Hip	Looking Away	Pointing w/ Hand on Hip
5	Tapping Fingers	Crossing Arms		Leaning Forward

<table>
<tr><th colspan="4">Positions</th><th>Effect</th></tr>
<tr><td colspan="4">11</td><td>t (default)</td></tr>
<tr><td colspan="2">21</td><td colspan="2">22</td><td>l (slide in from left)</td></tr>
<tr><td>31</td><td colspan="2">32</td><td>33</td><td>s (sink)</td></tr>
<tr><td>41</td><td>42</td><td>43</td><td>44</td><td>h (hop)</td></tr>
<tr><td colspan="4" rowspan="4">hide
(Technically not a position, but when used with t or l and zorder 1, will fade the character out. Make sure to use "hide <character>" in the next line after.)</td><td>f (focus)</td></tr>
<tr><td>i (instant appearance)</td></tr>
<tr><td>d (dip)</td></tr>
<tr><td>hf (hop + focus)</td></tr>
</table>

Expressions:

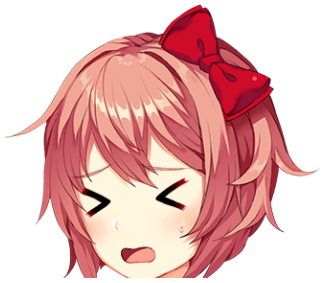
Expression	Sayori	Natsuki	Yuri	Monika
a				
b				

c				
d				
e				

f				
g				
h				

i				
j				
k				

l				
m				
n				

o				
p				
q				

r				
s				
t				

u				
v				
w				

x				
y				
z				

Sayori Special Pose Expressions (Only ones able to be used for this pose)			
a	b	c	d
			

Monika Special Pose Expressions (Only ones able to be used for this pose)	
a	b
	

Yuri Yandere Expressions (used as y<number>)						
1	2	3	4	5	6	7
						

Yuri Turned Away Expressions		Natsuki Special Expressions (Not compatible with pose 5, used as <pose>2<"b" if casual><expression letter>)	
a		a	
b		b	

c		c	
d		d	
e		e	

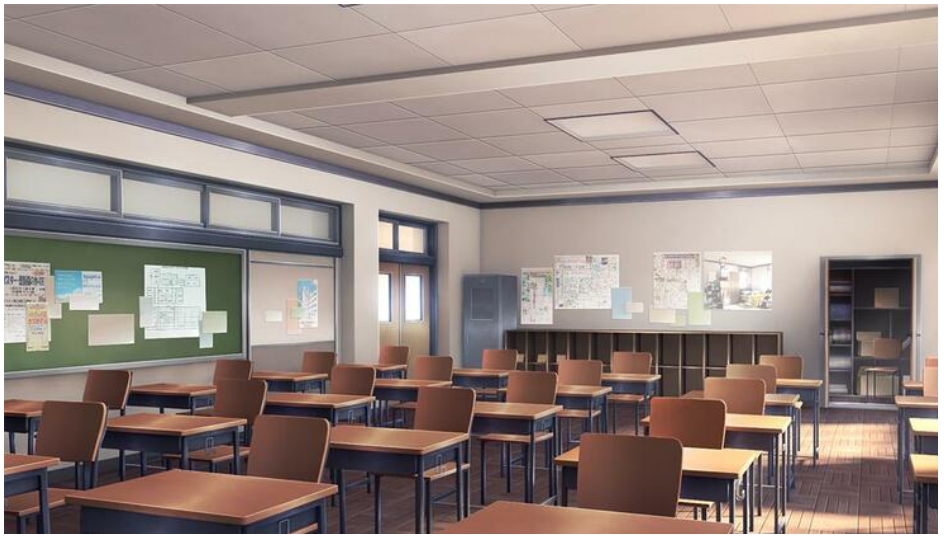
	f	
	g	
	h	

i	
These are not used like the above and are instead used as natsuki scream and natsuki vomit respectively.	
scream	
vomit	

Backgrounds:

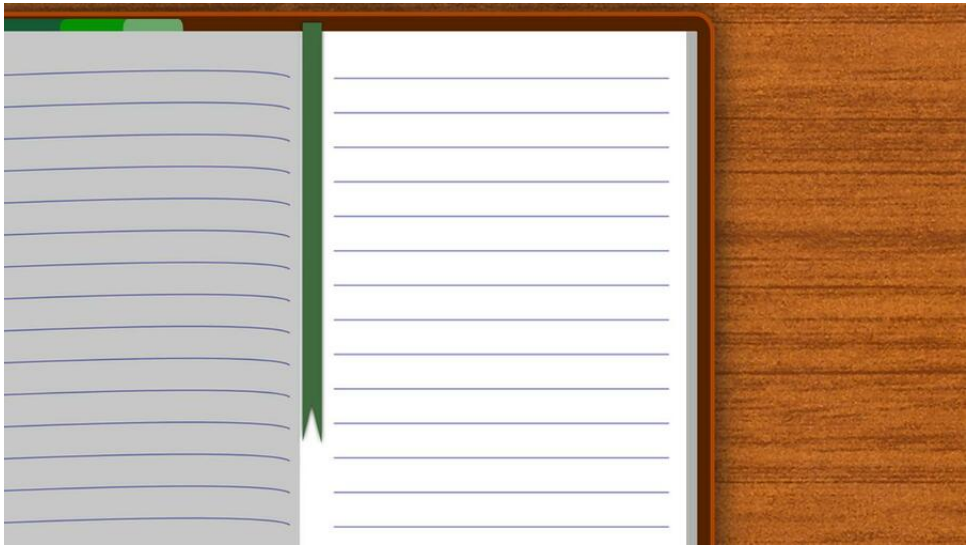
Backgrounds	
Names	Images
residential_day	

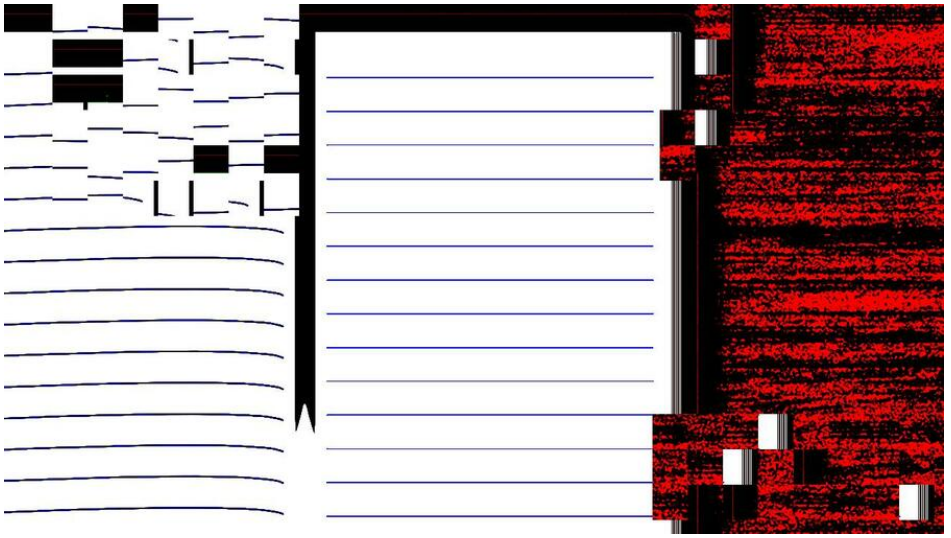
class_day	
corridor	

club_day	
club_day2 (⅕ chance to show, otherwise club_day will appear. If need be, you can change this to always appear in definitions.rpy by removing the “choice:”s)	

closet	
bedroom	

sayori_bedroom	
house	

kitchen	
notebook	

notebook-glitch	

Audio:

Audio			
Music		Sound Effects (Note: Only the first four are defined in definitions.rpy. If you want to play any other sound, use this format: play sound "sfx/<sfx name>.ogg"	
t1	Title Theme	closet_open	Opening Door
t2	Ohayou Sayori!	closet_close	Shutting Door
t2g	Ohayou Sayori! (Wobbly 4 Second Section)	page_turn	Page Turn
t2g2	Ohayou Sayori! (Rapid Glitch Noise)	fall	It's... a fall sound effect
t2g3	Ohayou Sayori! (Gradual Pitch Increase)	s_kill_glitch1	Glitching Sound
t3	Main Theme	fall2	A much rougher falling SFX
t3g	Main Theme (Off Key Notes)	giggle	An evil giggle.
t3g2	Main Theme (Start From Weird Note)	glitch1	Really weird descending glitching sound
t3g3	Main Theme (Reverb + Strange Wet Noises [Before Yuri Cutting])	glitch2	Another weird sounding descending glitch
t3m	Main Theme	glitch3	A jumping glitch
t4	Dreams of Love and Literature	gnid	Kinda sounds like when your

			ear is ringing. It's a pretty spooky buildup.
t4g	Static + Error Noise	interference	Triangleish Wave Error
t5	Okay, Everyone!	monikapound	The SFX playing when you delete Monika in Act 3
t5b	Okay, Everyone!	mscare	Monika's jumpscare
t5c	Okay, Everyone! (See Misc Notes for why Okay, Everyone has three separate defintions)	run	Natsuki neck snapped run (at you).
t6	Play With Me	slap	Slap!
t6g	Play With Me (Bitcrushed Melody Piano)	smack	More like a bump, but smack!
t6r	Play With Me (Sped-up + Reversed)	stab	Three slow, consecutive stabs
t6s	Play With Me (Yuri Death)	yuri_kill	The little music that plays when Yuri kills herself, featuring stab noises
t7	Poem Panic!	crack	Natsuki neck snap
t7a	Poem Panic! (First Melody Loop)	eyes	Super low pitched breathing(?)
t7g	Poem Panic (Act 2 Argument)		
t8	Daijbou!		

t9	My Feelings	
t9g	My Feelings (Harpsichord + Fast)	
t10	My Confession	
t10y	My Confession (Yuri)	
td	Sayo-nara (Note; includes stinger)	
m1	Just Monika.	
mend	I Still Love You	
ghostmenu	Ghost Menu Theme (Static + Weird Breathing)	
g1	Low Sawtooth Wave (Sounds like an error)	
g2	Lower Sawtooth Wave (Also sounds like an error, but worse)	
hb	Heartbeat	
end-voice.ogg	Monika's credits dialogue.Call this with: play music "bgm/end-voice.ogg"	
credits.ogg	Your Reality. Call this with: play music "bgm/credits.ogg"	

s_kill_early.ogg	Static that plays when you delete Monika.chr early. Call this with: play music "bgm/s_kill_early.ogg"	

Glitching:

There are a lot of glitch functions in DDLC. I mean, that is the core of the game. So, let's start out with the most notable ones:

To use the black outlined glitch font, do the following:

```
$ style.say_dialogue = style.edited #Glitch Font Starts

"<normal dialogue would go here>"

$ style.say_dialogue = style.normal #Glitch Font Ends
```

To use the long string of glitch characters, do the following:

```
$ gtext = glitchtext(<number of glitch characters you want>) #Any variable works here, the game just uses gtext
"<normal dialogue>[gtext]" #gtext is when the glitch text starts
```

For tearing up the screen (ex: like in Sayori's death or any glitch ever), use the following:

```
show screen tear(20, 0.1, 0.1, 0, 40)
play sound "sfx/s_kill_glitch1.ogg" #You can change this sound if you want
pause <however long you want the glitch effect to last, generally around 0.25>
hide screen tear
```

These parameters can actually be changed.

```
# show screen tear(number, offtimeMult, ontimeMult, offsetMin, offsetMax)
# "number" tells how many times the screen is divided. This makes the sections that glitch larger or smaller.
# A number is chosen between offtimeMult and ontimeMult to offset each section
# offsetMin and offsetMax tell how far the glitch will distort the screen. Setting the Max higher makes each
section move farther out
```

For darkening the corners of the screen (Vignette), use the following:

```
show vignette: #Add zorder 3 or higher to be above the characters
        alpha <value of transparency (1.0 for solid, 0.0 for completely transparent>

#You can also use "at vignettefade(start time after value in seconds)" to gradually fade in the effect over 25
seconds.
#"at vignetteflicker(start time after value in seconds)" will flicker the effect.
#The start times can be changed in transforms.rpy

hide vignette #End the effect
```

For the noise effect:

```
show noise: #Add zorder 3 or higher to be above the characters
        alpha <value of transparency (1.0 for solid, 0.0 for completely transparent>

#You can also use "at noisefade(start time after value in seconds)" to gradually fade in the effect over 5 seconds.
#The start times can be changed in transforms.rpy

hide noise #End the effect
```

For the zooming in w/ the heartbeat music, use this:

```
show layer master at heartbeat #Start

#Code

show layer master #End
```

Poems:

I should've added this much earlier but... woops. Creating and calling poems is actually relatively easy. I suggest first and foremost you make your own poems_mod.rpy or something to put your poems into, but you can also add on to the original poems.rpy from the base game if you'd like.

Now, to make poems... first, let's write one.

```
    poem_m_as1 = Poem(
    author = "monika",
    title = "Conscious",
    text = """\
I thought.
Thought.
Thought.\n
Struggled deeply.
The year was hard.\n
Friends.
Family.\n
Empty.\n
Silhouettes roaming the halls.
A blur.\n
And then.
Enter You.\n
Time passes.\n
And I,
the Epiphany.\n
My tears are stained red."""
```

A wonderful poem, right? (please don't use it for anything) But that's not to be discussed here. Let's break down what we just did here.

First, we defined what variable our poem is stored in. So, in this example, it would be poem_m_as1. You can name this whatever you want, but I suggest giving it the suffix poem_ at least.

Next, we determine the author. The author determines what font the poem will be using, except for Yuri's Act 2 fonts (we will get into those in a moment). This is simply what girl you want's name in all lowercase.

If you have a new character, you can first add a new font at the bottom of your script like so:

```
style <character>_text:
    font "<location of your font>"
    size <size of text>
    color "#000"
    outlines []
```

Next, you'll need to update a section of the original poems.rpy:

```
if currentpoem.author == "yuri":
   if currentpoem.yuri_2:
      text "[currentpoem.title]\n\n[currentpoem.text]" style "yuri_text"
   elif currentpoem.yuri_3:
      text "[currentpoem.title]\n\n[currentpoem.text]" style "yuri_text_3"
   else:
      text "[currentpoem.title]\n\n[currentpoem.text]" style "yuri_text"
elif currentpoem.author == "sayori":
   text "[currentpoem.title]\n\n[currentpoem.text]" style "sayori_text"
elif currentpoem.author == "natsuki":
   text "[currentpoem.title]\n\n[currentpoem.text]" style "natsuki_text"
elif currentpoem.author == "monika":
   text "[currentpoem.title]\n\n[currentpoem.text]" style "monika_text"

#Add this.

elif currentpoem.author == "<character>":
   text "[currentpoem.title]\n\n[currentpoem.text]" style "<character>_text"
```

And that should do it. Do keep in mind that you'll need an "Okay, Everyone!" variation for that character that you will likely have to call specifically with the showpoem function, unless you add a 5_<character>.ogg file in audio/bgm.

Now, for actually calling a poem. There are a number of things you can do here, but the base function is:

```
call showpoem (<poem variable>)
```

This will just show the poem as normal. The "Okay, Everyone!" variation will play and yada yada. But, there are actually a lot more variables you can mess with.

```
call showpoem(poem, music, track, revert_music, img, where, paper)

# Poem = Poem being shown
# Music = If True, it will change the music to either the Track variable or the "Okay, Everyone!" variation
# Track = The song you want to play instead of "Okay, Everyone!"
# Revert_music = Set to true if you want the variation of "Okay, Everyone!" to change back to the normal version
after you close the poem.
# Img = If you want to show an image (or character) in the background while the poem is on screen. Good for scares.
# Where = What position you want the image shown
# Paper = If you want a special image for the poem paper to be shown, like Yuri's piss paper
```

So, if I wanted to call Yuri's piss paper poem, I'd do...

```
call showpoem (poem_y23, track="bgm/5_yuri2.ogg", revert_music=False, paper="images/bg/poem_y2.jpg", img="yuri
eyes", where=truecenter)
```

Not too hard, right?

Special Poems:

This is actually really easy to do, but I feel the need to give it its own section anyways because it's pretty important.

All the special poems are simply defined in poems_special.rpy, I'd suggest adding onto that or making a duplicate poems_special_mod.rpy.

This is the basic label for a special poem:

```
image poem_special<number> = "<location of poem>"

label poem_special_<number>:
    $ quick_menu = False
    play sound page_turn
    show poem_special<number> with Dissolve(1.0)
    $ pause()
    $ quick_menu = True
```

There's nothing really else to say about this, other than you can really do whatever the hell you want with this. $ pause() pauses until you click, so if you want another image shown or a sound effect played or whatever, just put it after or before those.

Misc Notes:

If a character combination isn't in defintions.rpy, use this function to add it:

```
image <character> <pose><expression> = im.Composite((960, 960), (0, 0), "<expression location>" (0, 0), "<pose location>")
```

You can really name the variable whatever you want, but following how the others are set up makes life easier in the long run. I recommend adding it with the other expressions in definitions.rpy.

Natsuki's special expressions do not work with her special pose normally, you'll have to edit a new one.

(THIS EXPRESSION HAS BEEN REMOVED FROM THE FILES AS OF DDLC 1.1.1, YOU WILL NEED TO ADD IT AS A CUSTOM ASSET)

She has an unused expression in her "old2" folder in images.rpa named "4t.png":

Isn't it cute? You can implement it using the same image function above. Bear in mind it's specific to her special pose.

To show a character, use this function:

```
show [character name] [character pose and expression] zorder [character priority] at [effect][position]
```

To hide a character, use zorder 1. Else, you can have zorder set to whatever you want above 2. Having one character above another will let you make them appear in front of the other (ex: the scene in DDLC where Sayori stole a bite of Natsuki's cookie.)

Formatting Tricks:

You can find a full list of the formatting tricks Ren'Py uses here: https://www.renpy.org/doc/html/text.html

But for some ones that might be more useful for a DDLC mod...

{nw} - Used in dialogue. This will skip to the next line without player input.
{i}dialogue{/i} - Italics
{fast} - Jumps to that point in the text immediately.
*{cps=value}dialogue{/cps} - Changes the characters shown per second. You'll likely want something like {cps=*2} to double the speed.*
{w=value} - Pauses and waits for a value of seconds before continue on the dialogue.

Basic transition for starting/ending a scene:

There are multiple transitions that can be used that I have yet to document, so take this as a basic guide. For beginning a chapter:

```
stop music fadeout 2.0
scene bg <background you want to use>
with dissolve_scene_full
play music <music you want>
```

Transition to a new scene:

```
stop music fadeout 2.0
scene bg <bg you want>
with wipeleft_scene
play music <music you want>
```

And ending a chapter:

```
scene black
with dissolve_scene_full
return
```

The Mystery of "Okay, Everyone!" and how it plays the variations:

So, I don't entirely understand this yet, but from what I can gather...
The poem music is controlled directly in poems.rpy at the very bottom of the script:

```
label showpoem(poem=None, music=True, track=None, revert_music=True, img=None, where=i11, paper=None):
    if poem == None:
        return
    play sound page_turn
    if music:
        $ currentpos = get_pos()
        if track:
            $ audio.t5b = "<from " + str(currentpos) + " loop 4.444>" + track
        else:
            $ audio.t5b = "<from " + str(currentpos) + " loop 4.444>bgm/5_" + poem.author + ".ogg"
        stop music fadeout 2.0
        $ renpy.music.play(audio.t5b, channel="music_poem", fadein=2.0, tight=True)
```

A label is set up as showpoem. This is really for, well, showing the poem that MC is seeing. It checks the track, music, if it should reverse the music, image, where it should show at, the type of paper, blah blah. What's important is a little lower, at "if music".

What this does is first, get the current position of the song playing. It then checks what song is playing. If it's the one defined as "track" in showpoem, it'll change "t5b" to play from the current position of the song, loop at the normal spot, and the track defined as "track". Else, it'll do the same thing, but play the "Okay, Everyone!" of whose poem you're looking at. So, if I were viewing Natsuki's poem, it'd play her variation of "Okay, Everyone!"

```
    if music and revert_music:
        $ currentpos = get_pos(channel="music_poem")
        $ audio.t5c = "<from " + str(currentpos) + " loop 4.444>bgm/5.ogg"
        stop music_poem fadeout 2.0
        $ renpy.music.play(audio.t5c, fadein=2.0)
    return
```

If revert_music is true, after the poem is closed, it'll go back to the normal "Okay, Everyone!" at the current position, defined as 5c.

The variations are not defined by default. You can add them in just like any other music in defintions.rpy, like so:

```
define audio.tmonika = "<loop 4.444>bgm/5_monika.ogg"
define audio.tsayori = "<loop 4.444>bgm/5_sayori.ogg"
define audio.tnatsuki = "<loop 4.444>bgm/5_natsuki.ogg"
define audio.tyuri = "<loop 4.444>bgm/5_yuri.ogg"
```

You would then use tmonika and etc. as normal