A Guide to using the new poem game system

What is the poem game system?

The poem game system is basically the poem minigame you play in DDLC when the day ends where you write a "poem" for one of your club members.

What was changed?

A lot. Mainly the use of dictionaries, classes, and inheritance for poem words, the poem characters and more.

How do I use the new system?

There are many modules that are now in-use

PoemWord

This class defines the poem words which are either extracted from *poem_game/poemwords.txt* or manually for certain bugged words like those in Act 2-3. This system is still the same as it was in DDLC.

ChibiTrans

ChibiTrans is responsible for the doki's hopping around the game and jumping when their favorite word is picked. This is useful for adding a Chibi character to the game that does not have any appeal (like Monika in Act 3).

Chibi(name)

Chibi is responsible for storing appeal data during the poem game. It inherits <u>ChibiTrans</u> meaning it also contains hop information along with point data. This is useful for adding a Chibi character that has appeal factors (Sayori, Natsuki, Yuri). It takes in a name string.
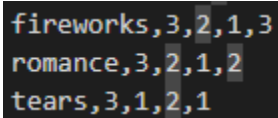
How do I call the new poem game?

Like before in DDLC, all you really need to do is do *call poem*

How do I add a new character?

This section assumes that you did not overly edit the source code or the poem screen.

1. Add another comma to the words in *poem_game/poemwords.txt* to dictate how your character likes said word. (Add any additional words if need be for said character.

    a.
    ```
    fireworks,3,2,1,3
    romance,3,2,1,2
    tears,3,1,2,1
    ```

2. Add another argument variable to the PoemWord class for this character and add a self variable inside the ___init___ function.

```python
class PoemWord:
    def __init__(self, s, n, y, l, glitch=False):
        self.sPoint = s
        self.nPoint = n
        self.yPoint = y
        self.lPoint = l
        self.glitch = glitch
```
a.

3. Make a variable of the character as either a *Chibi* character or *ChibiTrans* character.
   a. Remember that *Chibi* is meant to store appeal data and move around while *ChibiTrans* only makes the chibi sprite move around.

```python
chibi_m = ChibiTrans()
chibi_y = Chibi('yuri')
chibi_l = Chibi('libitina')
chibi_a = ChibiTrans()
```
i.

4. Edit lines *153-177*, to add your character image data for hopping in Act 1 and 2 (if applicable).

```python
# Act 1
if persistent.playthrough == 0:
    if t.sPoint >= 3:
        renpy.show("s_sticker hop")
    if t.nPoint >= 3:
        renpy.show("n_sticker hop")
    if t.yPoint >= 3:
        renpy.show("y_sticker hop")
    if t.lPoint >= 3:
        renpy.show("l_sticker hop")
else:
    # Act 2
    if persistent.playthrough == 2 and chapter == 2 and random.randint(0,10) == 0: renpy.show("m_sticker hop") #1
    elif t.nPoint > t.yPoint: renpy.show("n_sticker hop") #Since there's just Yuri and Natsuki in Act 2, whoever
    elif persistent.playthrough == 2 and not persistent.seen_sticker and random.randint(0,100) == 0:
        renpy.show("y_sticker hopg") #"y_sticker_2g.png". 1/100 chance to see it, if we haven't seen it already.
        persistent.seen_sticker = True
    elif persistent.playthrough == 2 and chapter == 2: renpy.show("y_sticker_cut hop") #Yuri's cut arms sticker.
    else: renpy.show("y_sticker hop")
```
a.

5. Add a line below before *progress +=1* that looks similarly like this.

```python
# Adds points to the characters and progress by 1.
chibi_s.charPointTotal += t.sPoint
chibi_n.charPointTotal += t.nPoint
chibi_y.charPointTotal += t.yPoint
chibi_l.charPointTotal += t.lPoint
progress += 1
```
a.

6. Before the *exec* inside of poem_game_finish, add a line that looks similarly like this.

a.
```
y_poemappeal[chapter] = chibi_y.calculate_appeal()
l_poemappeal[chapter] = chibi_l.calculate_appeal()

# Poem winner always has appeal 1 (loves poem)
exec(poemwinner[chapter][0] + "_poemappeal[chapter] = 1")
```

    i. Make sure you also defined this in *definitions/definitions.rpy*

1.
```
default y_poemappeal = [0, 0, 0]
default m_poemappeal = [0, 0, 0]
default l_poemappeal = [0, 0, 0]
```

7. Add your idle chibi image inside the poem label as such.
    a. You may change the transform of where it's at though.

b.
```
if persistent.playthrough == 2 and chapter == 2:
    show y_sticker_cut at sticker_right #Replace Yuri's sticker with the "cut arms" sticker..
else:
    show y_sticker at sticker_right #Yuri's sticker
if persistent.playthrough == 2 and chapter == 2:
    show m_sticker at sticker_m_glitch #Monika's sticker
show l_sticker at sticker_mid
```

8. Define your sticker images and animations below the image definitions comment.
9. Save, and run DDLC. If all is good, the game boots up, and your new character should appear and interact fine.

Credits

Thanks to Elckarow for feedback and a base idea for this rework