Ren'Py

create your own visual novel

Ren What?

Ren'Py is a visual novel engine that helps you use words, images, and sounds to tell stories with the computer. These can be both visual novels and life simulation games.

http://www.youtube.com/watch?v=YZiKwuokGMc

Creating a New Project

- Go to the folder renpy-6.12.1 and click the RenPy icon.
- Choose "New Project".
- Choose your folder.
- Choose "template".
- Choose a color theme (don't worry, you can change it later)

Tutorial

Launch Edit Script Game Directory Check Script (Lint) Choose Theme Delete Persistent Archive Files Build Distributions

Ren'Py

Select Project
New Project
Ren Py Games List
Ouit

Options
Ren'Py Help
Tutorial Game



- Click on "Edit Script"
- Your script and images should all be in the same folder as your script. To see where your script is saved, click on File, Save As and look at the Path.



New Project

Please type the name of your new project.

Artemis

Cancel

Launch Edit Script Game Directory Check Script (Lint)

Ren'Py

- Instead of functions, we have "labels" in RenPy.
- You still need to indent.Ex.

label start:

"First msg"
"Second msg"

A Simple Game

- Start by deleting everything in the script editor
- When entering text, remember to encapsulate it in double quotes. If you want double quotes to appear in the visual novel, you should put a forward slash before each one.

"Hi, \"The Phantom of the Artemis\"" will be displayed as:

Hi "The Phantom of the Artemis"

Go to RenPy and click "Launch" to see what your game looks like! (click to make the game go forward)

□ options.rpy X □ screens.rpy X script.rpy X label start: "Welcome to Artemis" "Me" "Um... What's Artemis?" "Only the coolest summer program EVER!" "Silence." "And then..." "Congrats! You've just created your first visual novel" "Me" "What is a \"visual novel?\"" 10

Characters

- You don't want you keeps typing the name of the character (in our example, "Me") whenever they speak.
- You can also choose a color for the name.
- BEFORE "label start," you can define characters like this:

Characters

define m = Character('me', color = "#c8ffc8)
 label start:
 m "blah blah haha"

Define a character for 'Me' and 'Durrah' and place them at their respective locations (use m and d)

Don't forget the single quotes for the name!

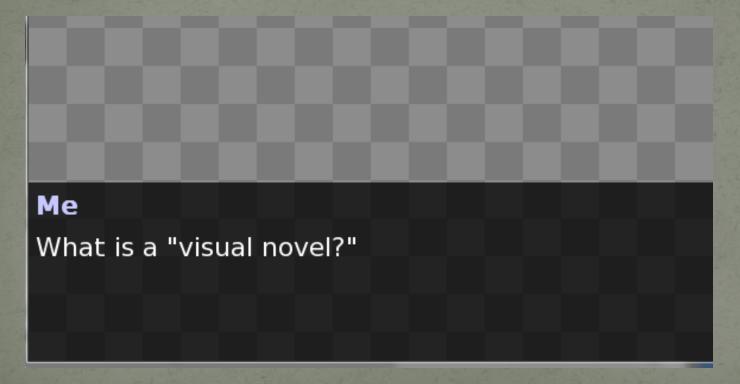
Don't worry about the color, there are reference tables online (you don't have to memorize the values :D)

Characters

Your code should look like this: (note the alignment)

```
□ options.rpy ×
               □ screens.rpy ×
                                       script.rpv X
define d = Character('Durrah', color="#c8ffc8")
define m = Character('Me', color="#c8c8ff")
A label start:
     d "Welcome to Artemis"
     m "Um... What's Artemis?"
     d "Only the coolest summer program EVER!"
     "Silence."
10
     "And then..."
11
     d 'Congrats! You've just created your first visual novel"
     m "What is a \"visual novel?\""
13
```

• And this is what you should see when you launch your game again:



Images (background)

- What kind of visual novel would this be without any "visuals?"
- If you store the images in your "game" directory inside your project's folder, you will just need to write the name and extension (like HTML) BEFORE label start:
 - image bg bu = "bu.jpg"
 - Image bg outerSpace = "space.jpg"
- You can use that in the background as a "scene" AFTER label start:
 - Scene bg bu= "bu.jpg"

Images (background)

• The first part of an image name is the image tag. If an image is being shown, and another image with the same tag is on the screen, then the image that's on the screen is replaced with the one being shown.

Images (background)

You are allowed to use spaces!

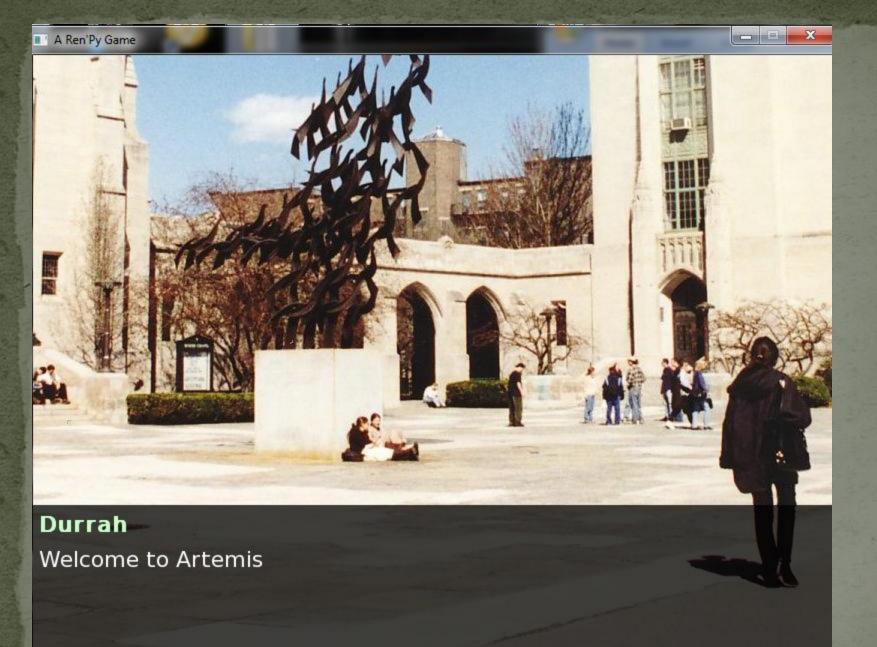
```
define d = Character('Durrah', color="#c8ffc8")

define m = Character('Me', color="#c8c8ff")

image bg bu = "bu.jpg"

label start:
    scene bg bu
    d "Welcome to Artemis"

m "Um... What's Artemis?"
```



What kind of pictures should I use?

• For the background: .jpg or .jpeg





For the foreground: .gif





Why does it matter?

 Saving the character in GIF format allows you to better integrate it with its surroundings, because GIF supports transparency.





Images (sprite)

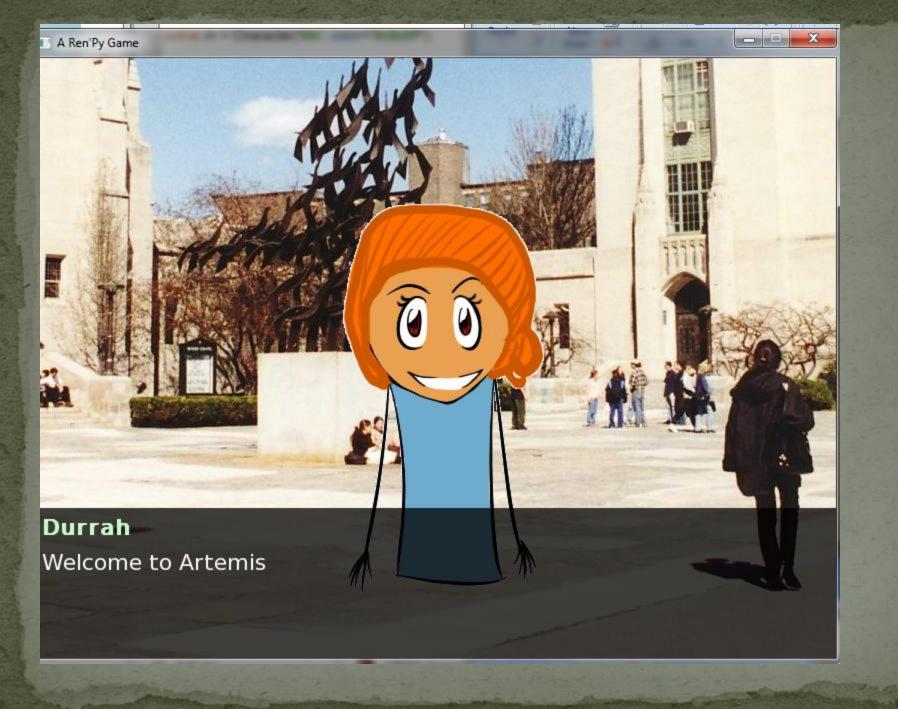
- You want to be able to see the characters, because that makes it more engaging! (And it's simple to implement)
- Declare the image like you did with the scene (BEFORE label start) and then make it "show".
 - image d happy = "durhappy.gif"
 - Show d happy

```
image bg bu = "bu.jpg"
image d happy = "durhappy.gif"
 label start:
   scene bg bu
   show d happy
    d "Welcome to Artemis"
    m "Um... What's Artemis?"
   d "Only the coolest summer program EVER!"
```

Images (sprite)

- Just like the scene can change from BU to outer space, the characters can go from happy to any other emotion.
- If you type a different "show x yy" a second time, It will overwrite the current one, and show you the new picture!
- You can also "hide" pictures (good practice) if you're making another picture show in the same location, or if the character is leaving.

```
s image bg bu = "bu.jpg"
 image d happy = "durhappy.gif"
 7 image d sad = "dursad.gif"
label start:
     scene bg bu
10
    show d happy
11
     d "Welcome to Artemis"
12
13
     m "Um... What's Artemis?"
1.4
     d "Only the coolest summer program EVER!"
15
16
     "Silence."
17
     show d sad
18
    "And then..."
19
     hide d sad
20
     show d happy
     d "Congrats! You've just created your first visual novel"
     m "What is a \"visual novel?\""
23
```



Positioning the Characters

Add an at-clause to a show statement. Simply say: show d happy at right

Or:

show d happy at left

20

21

22

23

24

hide d sad

show d happy at right

d "Congrats! You've just created your first visual novel" m "What is a \"visual novel?\""



Music and Sound

- Most games play music in the background. In Ren'Py, music files automatically loop until they are stopped by the user.
 - play music "illurock.ogg"
- When changing music, one can supply a fadeout clause, which is used to fade out the old music when new music is played.
 - play music "illurock.ogg" fadeout 1.0

- Music can be stopped with the stop music statement, which can also optionally take a fadeout clause.
 - stop music
- Sound effects can be played with the play sound statement
 - play sound "effect.ogg"
- Ren'Py supports formats other than .ogg for sound and music e.g. mp3, but make sure you check the copy rights before using the music.

Labels

- Like functions/methods (chunks of code you can call from anywhere in your program).
- "label" is a keyword for functions (like def).
- You must have noticed that we had a "label start" before the game code. It is a keyword that signals where the program begins. If you want a function to run, you will need to call it (or call a function that calls it).

Jumping to Labels

To call a label, you "jump" to it.

Keep your "start"
label at the top,
that is good
coding style and
makes it easier to
see how things
flow.

```
g label start:
     jump firstMeeting
L 11
12 label firstMeeting:
      scene bg bu
      show d happy
      d "Welcome to Artemis"
 16
      m "Um... What's Artemis?"
      d "Only the coolest summer program EVER!"
 19
      "Silence."
      show d sad
      "And then..."
      hide d sad
 23
      show d happy at right
      d "Congrats! You've just created your first visual novel"
      m "What is a \"visual novel?\""
 27
```

Jumping to Labels

- Remember to jump back to where you were if you want to continue executing the code that is after the jump.
- Ren'Py will NOT
 automatically jump
 back to the label you
 jumped from.

label start: jump firstMeeting

> jump blah "after blah"

label blah:

"blah" jump start

Menus

- The menu statement introduces an in-game-menu.
- It takes a block of lines, each consisting of a **string** followed by a **colon**. These are the menu choices which are presented to the user.
- Each menu choice should be followed by a block of one or more Ren'Py statements. When a choice is chosen, the statements following it are run.

A Game with Menus

Let's see what the play with the visual novel that now has a simple menu. (Code on next slide)

```
image d happy = "durhappy.gif"
image d sad = "dursad.gif"
image d superHappy = "dursuperhappy.gif"
image d thinking = "durthinking.gif"
```

```
hide d sad
 34
       show d happy at right
 35
       d "Congrats! You've just created your first visual novel"
 36
       menu:
\mathbf{v}_{37}
        "What is a \"visual novel?\"":
₹38
          jump teach
L 39
        "Awesome! I've always wanted to create one!":
40
          jump happy
 41
F 42

▼<sub>43</sub> label teach:

       hide d
```

```
43 label teach:
       hide d
 4.4
       show d thinking
 45
       d"I can explain the concept and teach how to create on"
 46
       show d happy
 47
       d "You'll love it!"
 48
       "Ending 1"
L 49
      return
 50
- 51

▼<sub>52</sub> label happy:

       hide d
 53
       show d superHappy
 54
       d "Great! Let's go create another one!"
 55I
       "Ending 2"
 56
       return
 57
```

If Statements

- After a point it becomes necessary to store the user's choices in variables, and access them again later.
- Based on the user's choices, the value stored in the variable changes, for example:
 - The number of answers you get right
 - True or False flags

```
label blah:

$ theCake = True

if theCake:

"Nope, the cake is a lie."
```

Ending the Game

- You can end the game by running the return statement, without having called anything.
- it's best to put something in the game that indicates that the game is ending, and perhaps giving the user an ending number or ending name.
 - d "Congrats! You've just created your first visual novel" m "What is a \"visual novel?\""
 - ".:. Good Ending." return

Your Visual Novel

- Design a game that has:
 - Three Characters.
 - Four backgrounds.
 - Three menus.
 - Five labels
 - Three different ending