Game Programming Workshop (part II)

What will you be doing?

Now that we created our first visual novel, we will be adding more complex things and make the game more interesting and interactive.

Asking for input

Let's start by making the game a little more personalizable for the player. We'll add a statement to ask the user to input their name. Then, instead of having a character named me, well create a Dynamic Character that will take the name that the user inputs at the beggining of the game.

```
6   define a = Character("Alex", color="#d2d2d2")
7   define m = DynamicCharacter("pname", color="#a6a6ff")
8
9   # The game starts here.
10
11  label start:
12
13      $ pname = renpy.input("What is your name?")
14      $ pname = pname.strip()
15
```

You can also use the player name in the middle of the dialogue by interpolating the variable. for this we use `[]`

```
a "Hello [pname]..."
```

Adding Transitions

Now let's add some transitions between scenes to make things more interesting.
RenPy provides many different transitions that changes the way our images and backgrounds appear on the screne. You can see the full list here

To call a transition we add the `with` statement and the name of the desired effect after the background.
For now we'll add a `fade` to the initial scene and a `dissolve` to the option scenes.

```
17      # Show a background. This uses a placeholder by default, but you can
18      # add a file (named either "bg room.png" or "bg room.jpg") to the
19      # images directory to show it.
20
21      scene bg coffee shop
22      with fade
23
```

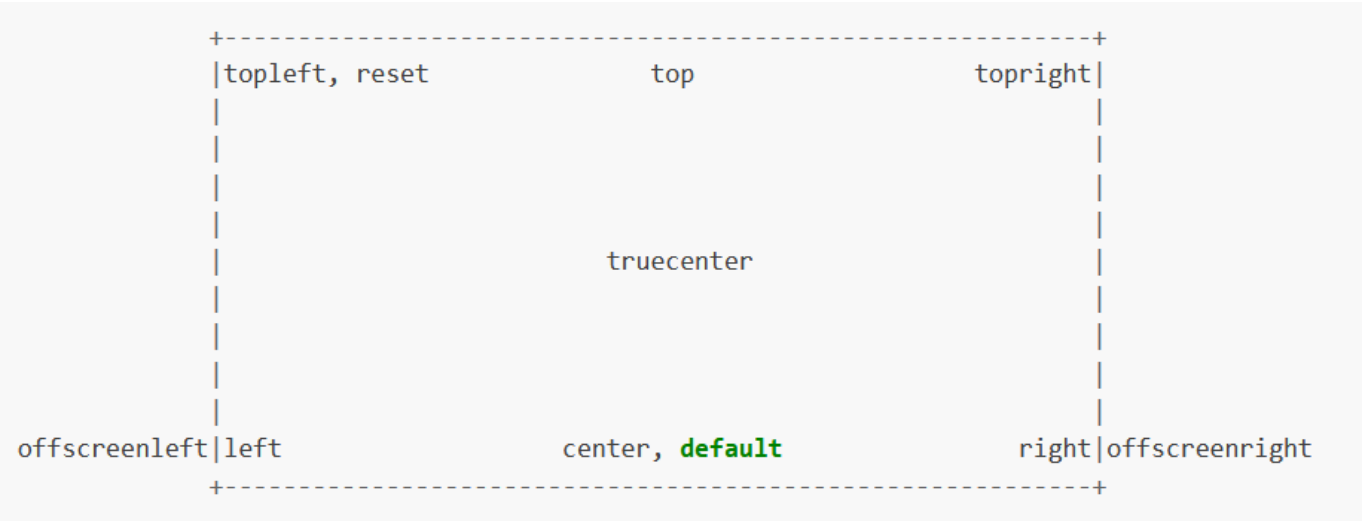
```
62      hide alex
63      scene bg street rainy
64      with dissolve
65
```

```
91      hide alex
92      scene bg street sunny
93      with dissolve
94
```

NOTE: RenPy also allows you to customize the pre-existing transitions or define your own. If you want to know more about it you can consult the documentation here.

Changing Character Position.

Imagine we have more than one character. We wouldn't want them to appear in the center all at the same time.
RenPy allows us to change the position of the characters in the screen. If no position is defined for the images the engine will use the default position. These are the pre defined positions in RenPy:



Let's change our game so that Alex appears on the right of our screen in the first scene.

```
26      # This shows a character sprite. A placeholder is used, but you can
27      # replace it by adding a file named "eileen happy.png" to the images
28      # directory.
29
30      show alex sad at right
31
```

We also need to change the position for the remaining sprites in this first scene.

```
58      show alex smile at right
59
```

```
87      show alex surprise at right
88
```

Now our game should look something like this:

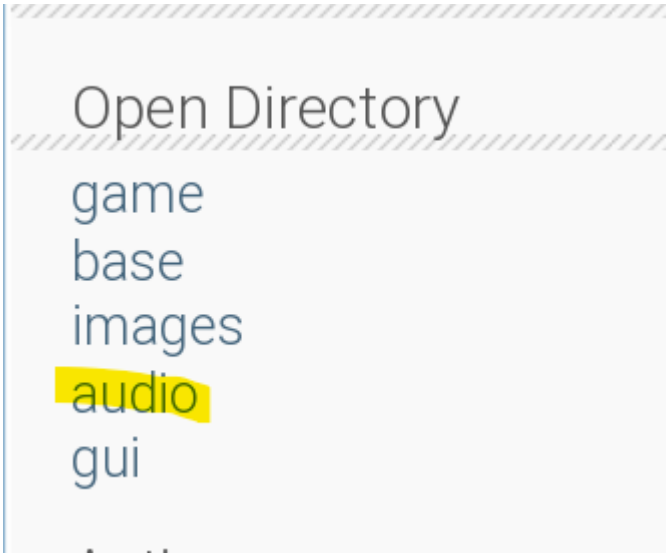

game

NOTE: Just like with transitions you can define your own coordinates for the character sprites by using the Transform function. You can consult the documentation about this here.

Adding music and sound effects

To add some depth to our game we can use sound effects.
RenPy allows us to add music, sound effects and voice tracks. This is done with the `play` statement.
By default, RenPy has 3 different sound channels that can play at the same time: `sound`, `music` and `voice`.
The engine also allows us to define whether we want the sound track to loop or not by using the keywords `loop` and `noloop`. In the first scene at the beginning of the game, our characters are in a coffee shop. To add to the immersion, let's add some backgound noise.

Start by adding the sound effects you downloaded to your *audio* folder. Then we'll add the corresponding sound effects to the three scenes of our game.



We want the sound effect to keep playing as long as the scene is ongoing, so we need to tell the engine to loop the track. We'll be adding the backgound noise to the music channel so that we can use the audio channel for other effects later:

```
21      scene bg coffee shop
22      with fade
23      play music "coffee shop.mp3" loop
24
```

Since we added an effect to the background in the other scenes, let's also add a fading effect to the sound track:

```
63      hide alex
64      scene bg street rainy
65      with dissolve
66      play music "rainy day.mp3" fadein 0.5 loop
67
```

If the track is too loud compared to the other background sounds we can also lower the volume:

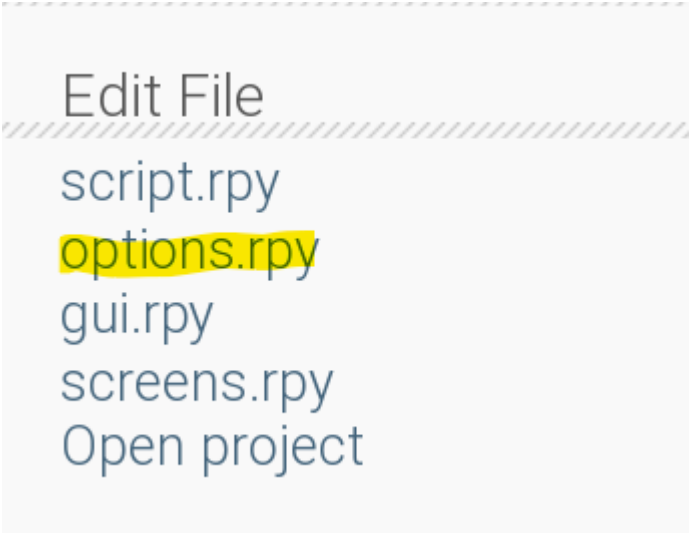
```
93      hide alex
94      scene bg street sunny
95      with dissolve
96      play music "sunny day.mp3" volume 0.75 loop
97
```

Now let's add a sound effect when the player chooses an option in the menu. This sound plays when the player clicks on either option, so we must add it to both:

```
44      menu:
45          "You decide to help Alex by telling them to..."
46
47          "Choose security.":
48              play sound "click.mp3"
49              jump security
50
51          "Pursue their passion":
52              play sound "click.mp3"
53              jump passion
54
```

Changing Configurations

Now we'll take a look at the configurations file.
To edit configurations we need to open the *options.rpy* script



Since we have created a new version of our game, let's start by changing the version.

```
24    ## The version of the game.
25
26    define config.version = "2.0"
27
```

Under the section Basics you can also find other options regarding the game. The `config.name` option allows you to change the title of the game without having to change the project name. You can also add a description to the about section using the `gui.about` option and decide if you want to show the title at the beginning by changing `gui.show_name` option.

Let's add a summary to our game:

```
## Basics ######################################################################

## A human-readable name of the game. This is used to set the default window
## title, and shows up in the interface and error reports.
##
## The _() surrounding the string marks it as eligible for translation.

define config.name = _("The Decision")


## Determines if the title given above is shown on the main menu screen. Set
## this to False to hide the title.

define gui.show_name = True


## The version of the game.

define config.version = "2.0"


## Text that is placed on the game's about screen. Place the text between the
## triple-quotes, and leave a blank line between paragraphs.

define gui.about = _p(""" This is a game about making decisions

Help you friend Alex chose the right job for them!

""")

```

Save your changes and test the game to see if the changes show up in the about section.

NOTE: You can also customize some parts of the GUI and default transitions, enable or disable sound, among other options. You can consult the documentation about the options file here

Custumizing GUI

The starting screen looks a little bland. Let's change the start menu and add a background image to our game.
First you need to add the background image to the *gui* folder so the game engine can find it. Then we'll also add a new image to change the in-game menu. To change the background and menu pictures, change the `gui.main_menu_background` and `gui.game_menu_background` options in the *gui.rp* file to the name of the new images:

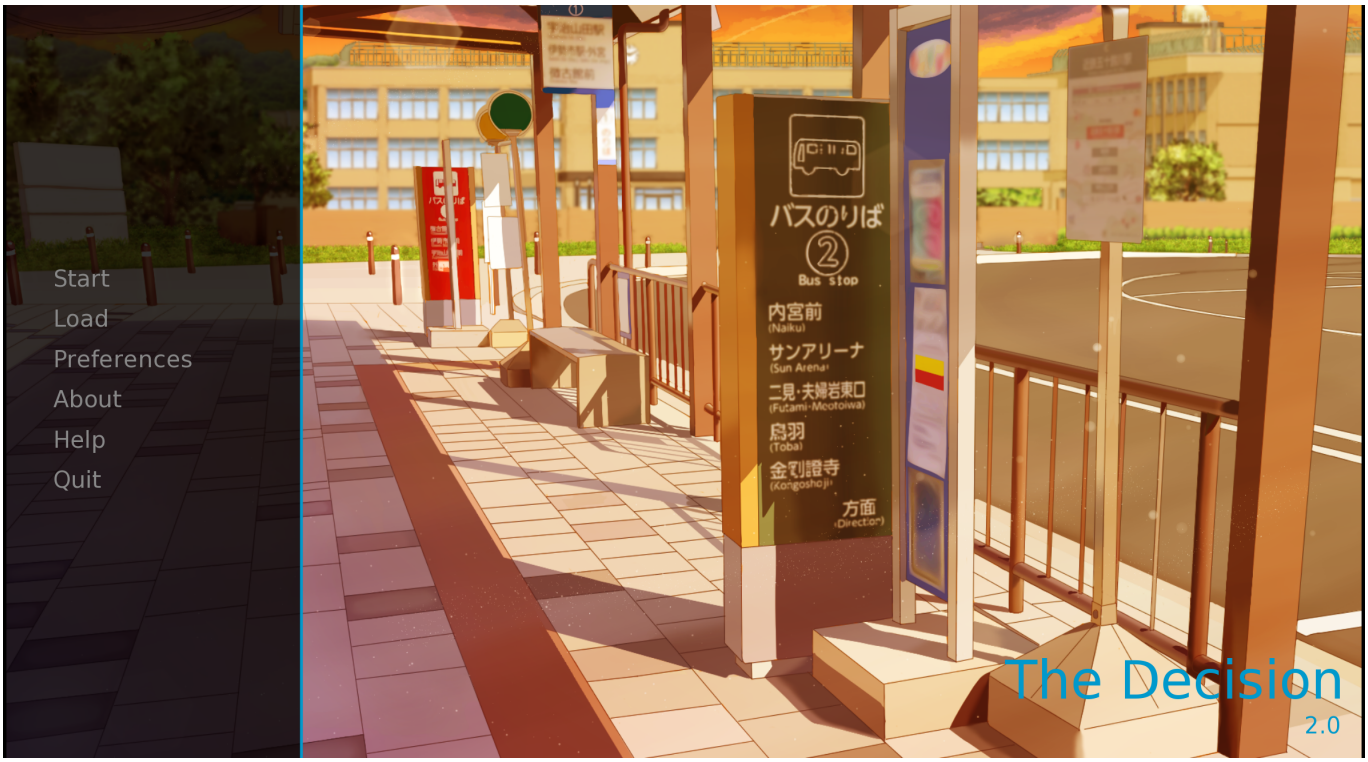
```
## Main and Game Menus #########################################################

## The images used for the main and game menus.
define gui.main_menu_background = "gui/street.png"
define gui.game_menu_background = "gui/new_game_menu.png"


```

NOTE: You can also change the aspect of the sidebar in the main menu by adding overlays. You can find the documentation about overlays here.

Releasing the Game

Now that we finished custumizing the game, we can release a playable version to the public.

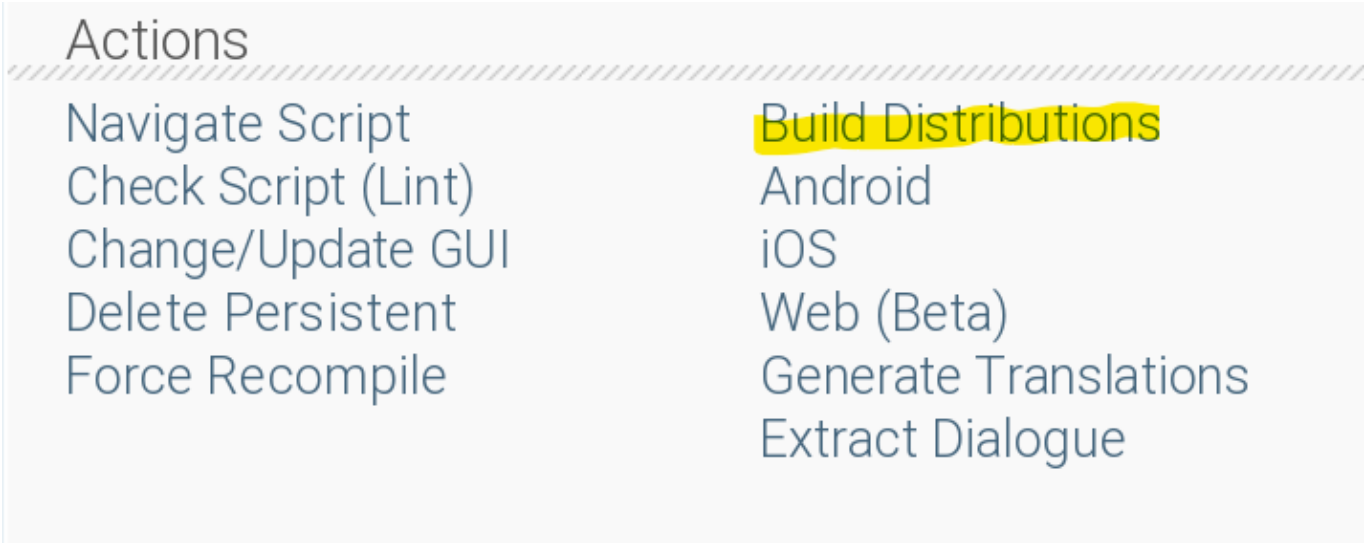
Let's start by checking for bugs by saving all the scripts and Launching the Project. It should look something like this:



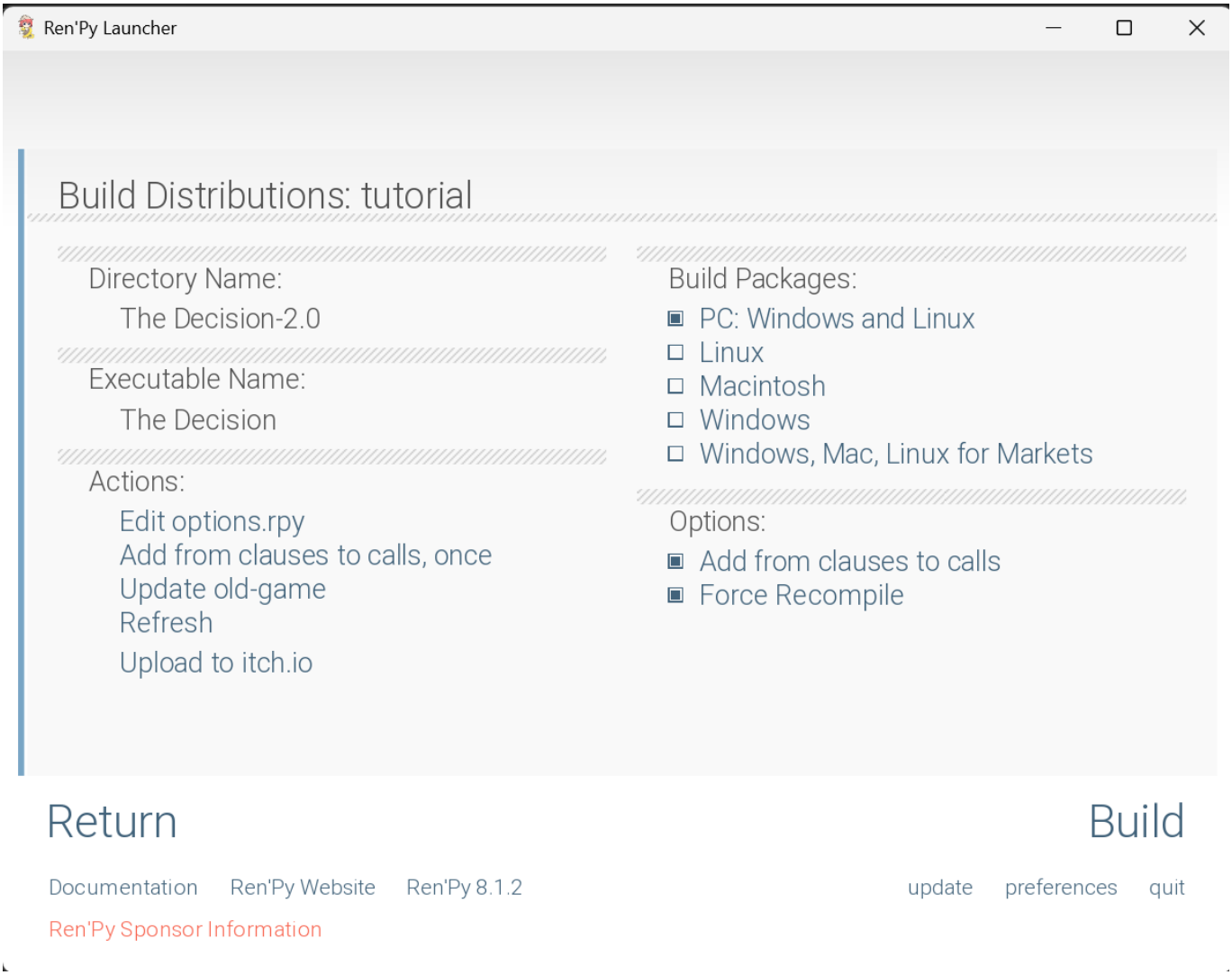
Then we can check more information by using the option *Check Script (Lint)* in the RenPy launcher. This will open a new script that contains the statistics and other useful information.

```
Statistics:

The game contains 30 dialogue blocks, containing 316 words and 1,766
characters, for an average of 10.5 words and 59 characters per block.

The game contains 1 menus, 0 images, and 24 screens.

Character statistics (for default language):
    * narrator has 12 blocks of dialogue.
    * m has 10 blocks of dialogue.
    * a has 8 blocks of dialogue.


Lint is not a substitute for thorough testing. Remember to update Ren'Py
before releasing. New releases fix bugs and improve compatibility.
```

After we make sure everything is correct, we can then select *Build Distribuitions* in the launcher, under the section *Actions*



Once you select the option the engine will open a new window:



Select only the *Pc: Windows and Linux* option, and leave all the other default options selected as is. Then click **Build** and the RenPy engine will start compiling and processing all the files. This usually takes a few minutes.

Once the engine finishes building the game it will open a folder that contains the .zip archive with your game. You can now share it with the world.