Handout

1. Preliminary Steps

Download Ren'Py onto a Windows computer: http://games.renpy.org/

Download Paint.NET onto a Windows computer: http://www.getpaint.net/

Website for creating characters: http://avachara.com/avatar/

2. Introduction

What will you learn?

In this lesson you will learn basic computer programming concepts as well as how to create a simple game with Ren'Py.

What is a computer program?

A computer program is a list of instructions that tell a computer what to do. Computer programs are written in a programming language, which is a language that allows you to get your computer to do certain things, such as display information on the computer screen or perform calculations. There are many programming languages out there. The following are some existing (and popular) programming languages: C, Java, PHP, Javascript, C++, Python, etc.[1] There are hundreds of programming languages out there. [2]

What is Ren'Py?

According to their official website,

> "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."[3]

A visual novel is an interactive fiction game which mainly contains still images. Visual novels are very popular in Japan. Many of the popular visual novels have been made into manga and anime.

In computer programming, an engine is a program that performs a core or essential function for other programs [4].

Do exercise 2.1.

3. Basic Programming Concepts

Variables

Variables in programming have the same function as they do in algebra: they store a value. That value can be a real number, an integer, or even values that aren't numbers, such as a letter, a word or a sentence.

In computer programming, the following basic types of variables exist:

- *Integer*: a number that is an integer, positive or negative.
- *Float*: a real number, which includes integer values as well as decimals and fractions.
- *Boolean*: either 0 or 1, no other values. This is in reference to binary numbers. In some programming languages, 0 is represented as the keyword 'false' and 1 as the keyword 'true'. The boolean type is generally used for 'if statements' to keep track of true or false.
- *Character*: a symbol that is not a number. For example, a letter is a character, the symbols $, %, @, #, are characters, as well as an apostrophe, a comma, and other solitary symbols on the keyboard.
- *String*: a sequence of characters. For example, a word is a string and a sentence is a string.

You can name your variables any name you want, as long as it conforms with the rules of variable naming. The naming rules for variables are as follows [5]:

- Variable names must begin with a letter of the alphabet or an underscore (_).
- After the first letter, variable names can also contain letters and numbers. They can't contain spaces or special characters (such as $, @, %, apostrophe, comma, etc.)
- Uppercase characters are different from lowercase characters. So, for example, a variable named 'DoG' is not the same as a variable named 'dog'.
- You can't use keywords from the programming language as variable names. Some keywords are 'while', 'for', 'if', 'else', etc. These can't be variable names on their own.
- Examples of valid variable names: Grade, GradeOnTest, Grade_On_Test
- Examples of non-valid variable names: Grade(Test), GradeTest#1, 3rd_Test_Grade
- It is good practice to have short, self explaining variable names. If your variable stores a string that is a friend's name, then having a variable such as CutePuppy would not be

helpful because, even though it is valid, it has nothing to do with the variable's value. A good variable name in this case would be NameOfFriend, or FriendName, or friendName, etc.

- It is also good practice to have short variable names. It is hard to keep track of and to keep typing long variable names, such as The_name_of_my_best_friend or OutputOfSigmoidFunction.

In some programming languages, you have to define variables before using them. This means that you have to specify their type and name first. It's not like this in all languages, for example, Python and PHP don't require you to specify the variable's data type.

In C++	In Python

int number1 = 5;
float itsafloat = 4.59;
bool isItAlive = 0;
char letter = 'A';
string message;
message = "This is a sentence"
	number1 = 5
	itsafloat = 4.59
	isItAlive = 0
	letter = 'A'
	message = "This is a sentence."

Variables in Ren'Py

Ren'Py variables are different from the basic variable types in other programming languages. This is because Ren'Py isn't itself a programming language, rather it is an engine made in Python (a programming language) to simplify the creation of visual novels and simulation games.

In this lesson, you will learn about 2 types of variables in Renpy, one for images and another for characters.

<u>Images</u>

Image variables are used to store the filename (or file path) of the image you want for either a character or a background. Later on in the code, these image variables are used for adding images to the game. The following is example of code for defining image variables:

image dog = "dog.png"
image teacher = "teacher.png"

```
image home = "home.jpg"
image classroom = "classroom.jpg"
```

The first two lines are character images corresponding to a dog and a teacher. The last two lines are background images corresponding to a picture of a home and a picture of a classroom. Notice how both cases correspond to different thing but are still defined in the same way: image variablename = “imagefilename”.

You display the image of a character by writing: show variablename. For example, to show the dog on the screen, you write: show dog. You display the image of a background by writing: scene variablename. For example, to show the classroom background on the screen, you write: scene classrooom. There are many other things you can do with these images, like have them move on the screen or hide them. You can view examples of these on the example program specified on the section “1. Preliminary Steps”.

Characters

Character variables are used to store the names of the characters in the story. You use the variable name as an abbreviation to refer to a character’s dialogue, without having to write their possibly long name over and over again. The following is example of code for defining character variables:

```
define dog = Character("Sparky")
define teacher= Character("Teacher")
```

The variable name used to refer to Sparky is dog, and the variable name to refer to Teacher is teacher. Example code for using these variables in dialog is the following:

```
dog "... ARF!"
(...)
teacher "Hello class. Your homework is due today. Time to turn it in."
```

Note that, for this lesson, character variables are only used for character dialog.

One last variable...

In RenPy, besides having RenPy specific variable types, you can also use normal variables. Since RenPy is based on the Python programming language, the syntax(grammar) for these

variable definitions is like that of Python. But there is a catch … the code written for these normal variables has to be preceded by a dollar sign ($). Here is an example:

```
$ game_score = 0
$ yourname = renpy.input("Hi! My name is ...")
```

Note: in RenPy, code that is preceded by a dollar sign is written in Python syntax. In other words, code that goes after a dollar sign is Python code.

Do exercise 3.1.

Control Structures

Control structures are:

1. If statements

The if statement allows you to control if a program enters a section of code or not based on whether a given condition is true or false. One of the important functions of the if statement is that it allows the program to select an action based upon the user's input.

For example, let's say that you have a variable corresponding to the score a player gets after playing a game you created. For simplicity, this score is either 0 or 1. A score of 0 means that the player lost while a score of 1 means that they won. We want to tell the player if they won or lost.

With if statements, the following algorithm would solve this problem:

```
if score == 0
  then player lost

if score == 1
  then player won
```

In this algorithm, if score is equal to 0, then the player lost. If score is equal to 1, then the player won. Notice how the condition after ‘if’ has to be true for the code ‘inside’ it to be achieved.

In programming, you can also add an 'else' to an if statement. Else indicates that whatever is in the 'else' will happen if the condition in the 'if' was not true.We can rewrite our example with an else:

```
if score == 0
  then player lost
else
  player won
```

What is ‘inside’ the else is achieve when score is not equal to 0, the opposite of the condition for the ‘if’. In general, else statements make code shorter and easier.

An example of if-else statements in human language is the following:

```
if 3 is greater than 50,
        then 3 is a big number
else,
        3 is a small number
```

Here, if 3 is not greater than 50, then it has to be less than or equal to 50.

Do exercise 3.2.

2. Loops

When you write a program, sometimes there is a piece of code that you would like to repeat over and over, for an unknown amount of repetitions. Determining the amount of repetitions, and writing the same code over and over again, is inefficient. For this we use loops. In computer programming, loops allow you to repeat a chunk of code for as many times as you want. The amount of times it is repeated depends on a condition that you specify (like in if-statements). In programming, while loops are the most common type of loop. The code within a while loop is repeated for as long as the condition beside the while keyword is true. Here is an example in pseudo code:

```
num = 0
while (num < 10)
    num = num + 1
```

In this example, the variable num is incremented by 1 until num becomes 10. So this while loop repeats the code ‘num = num + 1’ 10 times.

Control Structures in Ren'Py

Control structures in Ren'Py are not that much different from those in most programming languages. But there is one extra control structure that is unique to Ren'Py, called 'menu'. Here we will explain the menu control structure, but for if statements and while loops we will just show an example written in Ren'Py.

1. Menu

In Ren'Py, menus are the highlight of life simulation games and some visual novels. They allow a user/player to choose what they want to happen within the game, out of a list possible outcomes provided. In the example program specified on the section "1. Preliminary Steps", there is one part of the game where you are given an option to choose between lying about your homework or telling the truth. The following code is an abridged version of the menu code in the example:

```
you "*On no ... what should I do?*"

menu:
    "Lie.":
        "You lied … things happen…"
    "Tell the truth.":
        "You told the truth … things happen…"
```

You use the menu control structure by first writing 'menu:' and then writing your options with the right spacings. You can add as making options to choose from as you want, it doesn't have to be just 2. But it is not very good to put an excessive amount of options.

2. If statements

Example code:

```
if (game_score < 0):
        scene lost with Dissolve(.5)
        "Sad Ending..."
        "Today's lesson is ... do your homework on the computer. Save it constantly, off course."
else:
        #it's > or = 0, but only logically 1
        scene won with Dissolve(.5)
```

```
        "Happy Ending!"
        "Today's lesson is ... telling the truth is the way to go!"
```

3. Loops

Example code:

```
you "This has to be a nightmare ..."
"You close your eyes and start to count."

$ count = 10
while count > 0:
        you "[count]..."
        $ count -= 1

You open your eyes and look around you."
you "...Darn it, it's real."
```

Comments

Comments in computer programming are lines of text that are ignored by the computer. They are used to make code more readable and clear, by allowing a programmer to write descriptions and explanations of parts of their code within their code. Comments aren't interpreted as code, so they can be any sort of text. They do not follow any particular syntax. Different programming languages have different ways of specifying what parts of code are comments. For example, in C++ any line that follows two slashes (//) is a comment, and in Python any line that follows a hashtag (#) is a comment. Here are examples of comments mixed with code:

In C++	In Python

//This is a comment.
 	
int Num = 4; //this is a variable named Num
	#This is a comment.
	
	Num = 4 #this is a variable named Num

Sense Ren'Py was written in Python, comments in Ren'Py are preceded by a hashtag (#) as well.

Do exercise 3.3.

Getting User Input

To make a program interactive, it usually needs to take in user input. This means that part of the data within the program will be determined by what the user writes or clicks on the screen (when the program is already running). Ren'Py menus can be considered as allowing user input, but in this case the user only clicks one of certain options. You can make many different types of applications, and give the user a certain amount of freedom, by allowing them to write input that will determine the values of certain things within the program (such as character names). Here is an example of how to get written user input in Ren'Py so to have a customizable character name:

```
$ yourname = renpy.input("Hi! My name is ...") #get user user input, will be name of protagonist/player
$ yourname = yourname.strip()                  #strip() removes blank spaces from yourname
```

Do exercise 3.4.

Errors and Debugging

When you write a program and try to run/launch it, you may have errors that stop your program from running correctly. It may either run but not the way you expected it, or not run at all. When it doesn't run the way you expect (doesn't give correct results, or the images don't move as you hoped, etc.), it is called a semantic error (which refers to problems in the logic of your program). When it doesn't run at all, it is most likely a syntactic error (which refers to not having written the code with the correct syntax/grammar).

Debugging is the term used when you try to fix errors in a program. There are many techniques for debugging, but we won't discuss those here. For now, if you get errors, make sure that you have written your code correctly. Ren'Py will output an error message and it should give you insight on what the problem is. If you don't understand the error message, then you can google it.

You can see what happens if you have errors in Ren'Py by intentionally adding errors into your program and reading any error messages that you may get or seeing how your program works (or doesn't work) with the error.

5. Exercises

2.1 Create an empty project in Ren'Py. Open the program by double clicking the executable called "renpy". Then, once the Ren'Py application has launched, click the "+ Create New Project" button. Follow the steps to create your empty Ren'Py project.

3.1 Declare a 'character' variable, an 'image' variable for your character, and a background image variable in your Ren'Py project. You may use any of the images provided in the demonstration project folder, as well as images you find on the internet that are free to use under public domain. See section "Programming in Ren'Py" if you get confused.

3.2 Read the following if/else statement, and perform the task it asks you to do:
if your character's name in exercise 3.1 starts with letters a through m:
    your character in your Ren'Py story must fight 50 duck-sized horses; write the speech he/she gives before going out to face this formidable force
else your character's name in exercise 3.1 starts with letters n through z:
    your character in your Ren'py story must fight 1 horse-sized duck; write the speech he she gives before going out to face this formidable foe

You should use the following syntax to have your Ren'Py character say something:
character_name "words_of_the_speech"

3.3 Practice using comments by adding some comments in your own Ren'Py project. Use the pound symbol "#" to start a comment. You can type anything you want in your comments!

3.4 Practice getting user input in your Ren'Py program. Ask your user to input whether your character's speech was good enough to win the fight against the 50 duck-sized horses or 1 horse-sized duck. Use the following syntax to do this:

```
$ result = maybe
$ result = renpy.input("Was this speech motivational enough to win the fight? Enter yes or no.")
#input
$ result = renpy.strip() #remove spaces
```

After you have done this, you may use an if/else statement and the value of the result variable to tell the user what happened to your character after the battle. Let a classmate play your Ren'Py game.

6. Extra Resources

- Programming Languages: http://en.wikipedia.org/wiki/Programming_language

- Ren'py website: http://www.renpy.org/
- Python's website: https://www.python.org/
- Ren'Py games created by others: http://games.renpy.org/

7. Appendix

Reading this is optional. This appendix includes some extra information for if you want to learn more about the world of programming. If you want to learn more about programming, programming languages or Ren'Py specifically, or if you have any questions about the topics from this lesson, I recommend that you use a search engine (I usually use Google) to find out more information about these things. There are many resources online on programming language.

Computer Science Terms

User, client, player	Person that uses the program/software/game.
Human Language	Language used by humans to communicate with each other. It includes any language, such as English, Spanish, Chinese, etc.
Computer Language	Language understood by the computer. This includes machine language (which consists of zeros and ones) and a programming language.
Code	Program instructions written for a particular programming language.
Algorithm	List of instructions to solve a problem, written in human language. Some examples of algorithms are: a recipe, a to-do list.
Pseudocode	It is a mixture of an algorithm and code. It is still an algorithm, but the instructions are closer to code syntax.
Syntax	Has to do with the 'grammar' of code. A syntactic error means that something is misspelled in the code, or not written correctly. Common syntactic errors are missing semicolons (in C++ and Java) as well as missing parentheses in equations (usually has to do with not adding closing

	parenthesis).
Semantics	Has to do with the meaning/logic of code. A semantic error means that the code isn't giving the expected results/output.
Input	In algebraic terms, for an equation y = x + 2, x is the input. So, the input is what you use to get the result/output (in the program).
Output	In algebraic terms, for an equation y = x + 2, y is the output. So, the output is the result that you get (from a program).

Debugging Tips

Since RenPy was written with Python, Ren'Py's syntax is very close to that of Python's. So, you should know that in Ren'Py(as well as in Python) indentations and newlines are important for the program to run without syntax errors. So make your you add tabs and newlines when necessary.

Remember to read error messages from Ren'Py, if any appear. The error message should mention the line number where the error is, and insight on what is wrong. (This technique mainly works with syntax errors). If you don't understand the error message, you can always put it into google (or any other search engine) and see if you can find any websites where other people have asked about the error (you should specify the programming language in the search box as well).

References

1. "Programming Language Popularity." *Programming Language Popularity*. N.p., n.d. Web. 27 Feb. 2015. http://langpop.com/
2. "List of Programming Languages." *Wikipedia*. Wikimedia Foundation, n.d. Web. 28 Feb. 2015. http://en.wikipedia.org/wiki/List_of_programming_languages
3. "The Ren'Py Visual Novel Engine." *Ren'Py*. N.p., n.d. Web. 28 Feb. 2015. http://www.renpy.org/
4. "What Is Engine? - Definition from WhatIs.com." *WhatIs.com*. N.p., n.d. Web. 28 Feb. 2015. http://whatis.techtarget.com/definition/engine
5. "Data Basics - Naming Variables." *Data Basics - Naming Variables*. N.p., n.d. Web. 28 Feb. 2015. http://mathbits.com/MathBits/CompSci/DataBasics/naming.htm