Coding in Gaming User Manual

A *very* basic language done by great people.

1. **Variable**

Coding in Gaming(Pronounced as CiG, from hereon), requires that variables declared must **always** start with a small letter. Variables declared follow the order shown below:

varname = value/expression.

CiG supports dynamic type binding, that is, you do not have any need to indicate a variable’s type.

1. **Input/Output**

CiG allows input of user and allows output of variable values and integers to output stream.

Use Accept(<variable name>) to request input from user and store in specified variable

Use View (<variable>/<integer>) to output variable’s value or integer.

1. **If/Else Statements**

CiG implements if-else statements in the form of Good and Bad endings(similar to Good/Bad end routes in visual novels.)

Good represents the if statements, and Bad represents the else statement.

They are invoked by using Good, followed by the condition needed to satisfy it enclosed in parentheses, followed by a Bad to represent the else statement.

The terms “PRESS\_START , and GAME\_OVER” are used to enclose code to be used in both if/else statements(think of them as C’s {})

Good ( condition )

PRESS\_START

code when if statement is satisfied

GAME\_OVER

Bad

PRESS\_START

statements

GAME\_OVER

1. **Control Structures**

CiG implements an if-else system,with support for nested structures.

Through the use of Good( condition ) and Bad( condition ), pertaining to good and bad endings shown on visual novels.

CiG also supports looping, in the form of the word Spam(pertains to the term of using the same move over and over again.)

Spam( condition to loop)

Control Structures’ statements are enclosed with the terms “PRESS\_START” and “GAME\_OVER”, signaling the start of a structure and the end of said structure.

1. **Functions**

Functions are written in the form of quests, invoking the term “Quest”, the function’s name, and the arguments used for it.

Calling a quest requires the user to invoke the term “Start\_quest”, along with the function name and the arguments needed.

Quest <identifier> (<parameter>) definition

Start\_quest <function\_name> (<parameter>) call

1. **Others**

PLY (Python Lex and Yacc) must be installed to run the source code.

<variable name> = <value> (Only integers are allowed)

Variables start with a lower-case letter and can be followed by alphanumericals or an underscore

Input/Output

Accept(<variable name>) (Accepts input from user and stores in variable inside parenthesis)

View (<variable name>/<value>) (Prints to the stream/Only allows variable and integers for output)

Control Structures

Good (expression) (Must evaluate to true) (equivalent to if-else structure)

PRESS\_START #Block Start

Statements

GAME\_OVER #BlockEnd

Bad

PRESS\_START

Statements

GAME\_OVER

All Control structures must have both block before statements

Spam (expression) loops until expression is not satisfied

PRESS\_START

Statements  
GAME\_OVER