

Villanelle User Guide

POEM Lab

Written and edited by:

**Sasha Azad
Owais Iqbal
Dr. Chris Martens
Tony Mosolf**

**North Carolina State University
Games As Visual Narrative Research Group**

Last Updated: May 5th, 2019

Table of Contents

Summary of Villanelle	2
Authoring with Villanelle	2
Onboarding Walkthrough Example	4
FAQ / Troubleshooting	9
Provide Feedback	9
Villanelle Development Team	10

Summary of Villanelle

(Quick description of Villanelle, its purpose, and intended uses.)

Villanelle is a project from the POEM (Principles of Expressive Machines) research lab at North Carolina State University. The project began in 2016 and has been worked on by a variety of graduate, masters, and undergraduate students in conjunction with Dr. Chris Martens. Since its inception, the undertaking has evolved and continues to be an ongoing process of improvement based on the feedback of developers like you. In its current form, Villanelle is a stand-alone application with a graphical interpretation of the created behavior trees for the author.

Villanelle's purpose is to be an authoring tool for autonomous characters in visual narrative games. It's primary goal is to reduce the amount of technical expertise required by the author to craft their AI and behavior trees. The system utilizes goal-driven behavior for its AI and hopes to provide a framework for the decomposition of goals and tasks into pre-established functions that the author can employ.

Authoring with Villanelle

Glossary:

Agent: An npc actor within the game

Conditions: The requirements for an action to be taken

Effects: The results of a particular action

Effect text: The text that will be displayed to the user after an action is taken

Effect tree: A behavior tree that is executed on the selection of an action

Selector: A behavior tree node that will attempt to execute the first successful action, unless all fail.

Sequence: A behavior tree node that will attempt to execute all of its actions in order, unless one fails.

Ticks: Representation of the amount of turns an action will take

User Interaction Tree: A behavior tree responsible for displaying scenes and actions to the player

User action: Opportunity for the user to interact with the game and make a decision

Authoring Structure:

Tree structure Explained:

Tree name: (Name of tree, as it will appear in the tree visualization)

- (Hyphen denotes that this entry is a member of the above's array)
- selector/sequence: (Choice of how the following "nodes" will attempt to be handled, should always be an array)
- description: (String to be displayed on the screen)
- effects: (Always an array of changes to the game state)
- effect text: (String to display on execution of the parent's action)
- user action: (Always has action text and effect tree)
 - action text: (String to describe the action that will be taken)
 - effect tree: (A behavior tree that is executed on the selection of the action)
- condition (Property that can be applied to any node, if the condition is false, the node returns failure)

Script Structure Explained:

Initialization: (*Required*)

- Variable := value
- ...

Agent Name: (*Optional*)

Condition: (*Optional*)

Selector/Sequence:

- Condition:
 - selector/sequence:
 - condition: variable == value
 - Effect text: "text"
 - Effects:
 - variable := value
 - ...

...

User Interaction: (*Required*)

- selector/sequence:
 - condition: variable == value
 - description: "text"
- condition: variable == value
- selector/sequence:
 - description: "text"
 - effects:
 - variable := value
- description: "text"

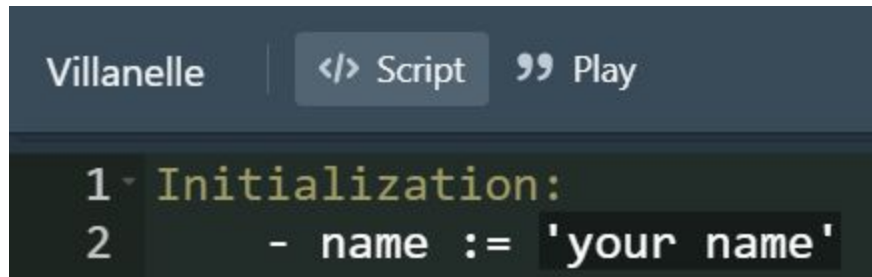
Onboarding Walkthrough Example

Hello World

Part 1: The Bare Essentials

In this first example, we'll be walking you through the creation of a very simple program. The very first thing you need to do is ensure that you have a blank editor. If the default text is there, simply remove it all. Next we need to set up our initialization section. This is where you establish the variables that will be used in the program.

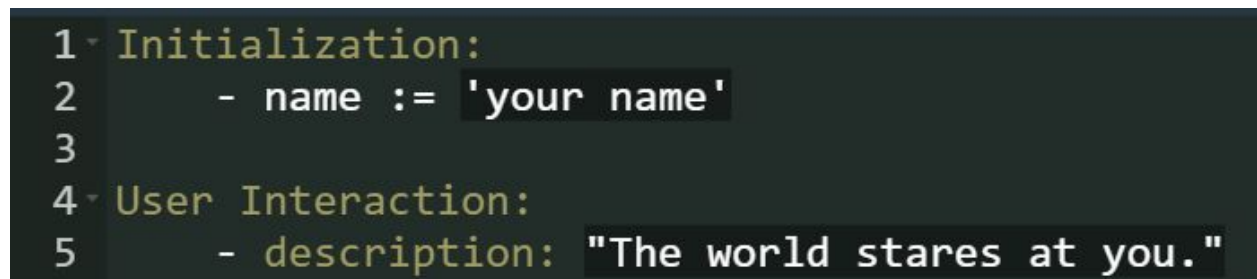
To begin this example we only need to initialize one variable. On the first line type "Initialization:". On the second line, add your name as a variable like this, "- name := 'your name'". Make sure your name is enclosed with single quotation marks! When you're done with this, your screen should look like the following:



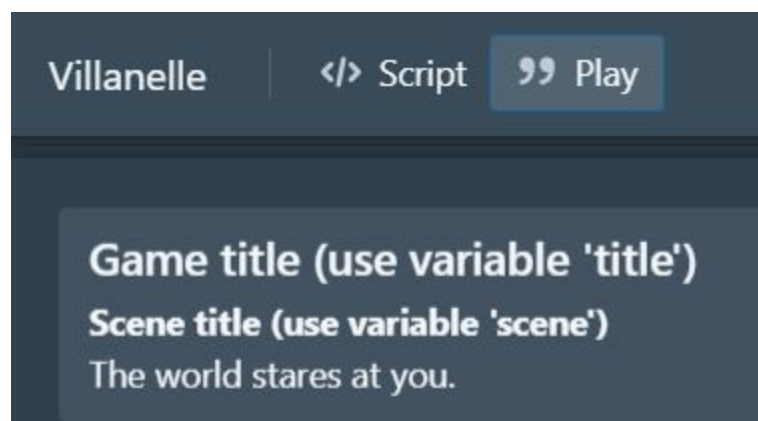
```
Villanelle | </> Script " Play  
1 Initialization:  
2   - name := 'your name'
```

Make sure that your lines are indented properly, the Villanelle application uses a indentation-based syntax, your indented line is a member of Initialization's array.

The next step is to set up the user interactions. In later examples this will be expanded into behavior trees, but for now we'll keep it simple. On the fourth line, type "User Interaction" with no indentation beforehand. This section will denote both the text displayed on the screen and the options that will be displayed to the user. Just like before with the variables, the next couple of lines will be indented with a hyphen in front of them. For line five, type '- description: 'The world stares at you.''. The description is the text that will be displayed to the user on the screen at the specified point in the game. We're now at the point where we can actually run our game. Below you'll see images of the expected current states of the editor and play tabs.



```
1 Initialization:  
2   - name := 'your name'  
3  
4 User Interaction:  
5   - description: "The world stares at you."
```



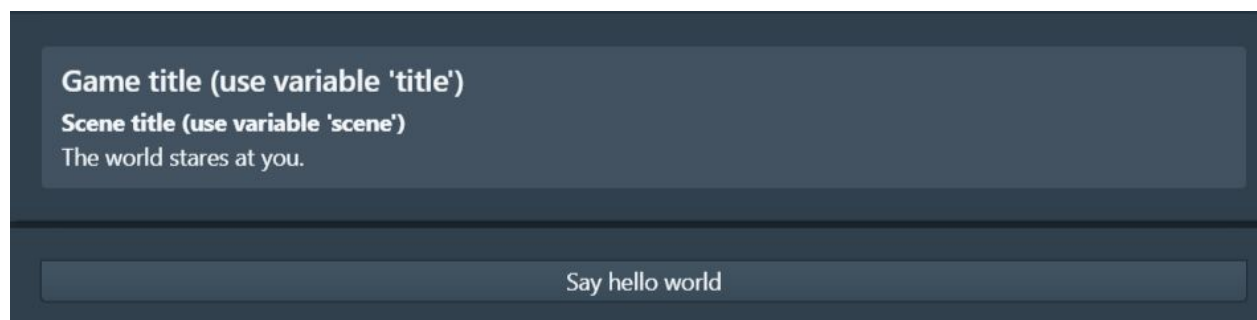
Part 2: The Basic actions

This is a nice start, but a game requires some interactivity. Next we're going to add some user actions. User actions represent opportunities for the user to interact with the game and make decisions. On lines six through nine of the editor, enter the following:

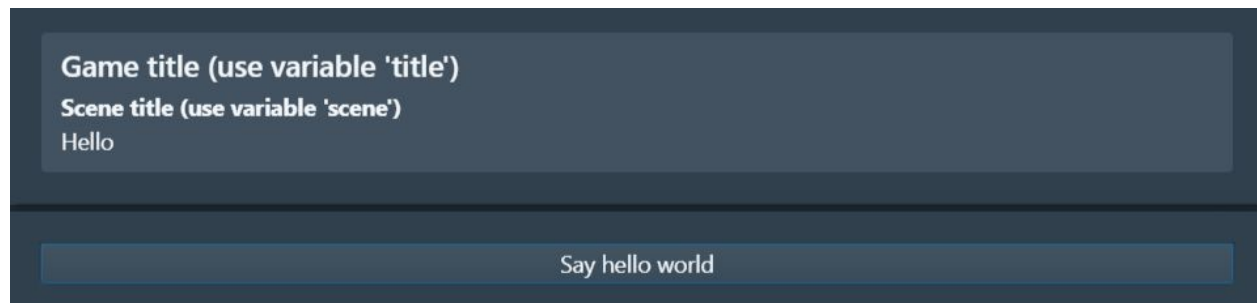
```
- user action:  
  action text: 'Say hello world'  
  effect tree:  
    effect text: 'Hello'
```

For this part of the example, the only thing you need to concern yourself with are the texts. The action text will be displayed on a button that the user can press and the effect text will be displayed on the screen when the user selects the action. Later in the example we'll elaborate on these other fields and how you can use them to change the state of the game. For now, you've completed your first program in Villanelle. Below are the expected displays of the play and editor tabs if you need to troubleshoot.

```
1 Initialization:  
2   - name := 'your name'  
3  
4 User Interaction:  
5   - description: "The world stares at you."  
6   - user action:  
7     action text: "Say hello world"  
8     effect tree:  
9       effect text: "Hello"
```



(Before pressing button)



(After pressing button)

Now that we've got our program compilable, we can start adding some more functionality to it. First let's improve the program by creating a second user action for saying goodbye to the world. The process is the exact same as before, we'll just have different text. Once you've completed that, you will have two different buttons to press and the displayed text on the screen will alternate between your two effect text strings.

Part 3: Using Variables

This is all well and good, but what if you want to use variables? We'll now begin the final part of this example, getting the game to display text with your name included! To do this, we will be using our initialized variables in our text. First, jump back up to your initialization section and add two new variables: `sayHello` and `sayGoodbye`. Set them equal to "The world says hello to" and "The world says goodbye to", respectively. Now we need to edit the effect texts. Instead of writing out a string, we can just provide the variables to it. To reference a variable, you include the `$` modifier in front of it. Therefore, the new effect texts will be `"$sayHello $name"` and `"$sayGoodbye $name"`. You can see the resulting code below and run it yourself to have a simple conversation with "the world".


```

1 Initialization:
2   - sayHello := 'The world says hello to '
3   - sayGoodbye := 'The world says goodbye to'
4   - name := 'your name'
5
6 User Interaction:
7   - description: "The world stares at you."
8   - user action:
9       action text: "Say hello world"
10      effect tree:
11          effect text: "$sayHello $name"
12   - user action:
13       action text: "Say goodbye world"
14       effect tree:
15           effect text: "$sayGoodbye $name"

```

Now we just have two things left to do before you understand all of the basics, set up our user interactions to edit variables, and create an AI tree that will use them. First we'll go in and edit our user actions. Instead of using effect text, we'll be using effects. Effects are an array of changes to the game state, so within that array we'll be able to modify our variables. For this example let's convert our sayHello and sayGoodbye variables to booleans that we'll initialize to false. Then let's edit our user actions so that both of them set their respective variable to true. Double check that your script looks like the following before moving on.

```

1 Initialization:
2   - sayHello := false
3   - sayGoodbye := false
4   - name := 'your name'
5
6 User Interaction:
7   - description: "The world stares at you."
8   - user action:
9       action text: "Say hello world"
10      effect tree:
11          effects:
12              - sayHello := true
13   - user action:
14       action text: "Say goodbye world"
15       effect tree:
16           effects:
17               - sayGoodbye := true

```

Part 4: Basic AI Tree

Now that you've gotten the hang of variables it's time to learn the most important functionality of Villanelle, writing AI behavior trees! For this example we'll be keeping it simple, our new tree will be called World and it will just display text like we've already been doing. To start, we'll make the new tree by writing its name with no indentation on a new line. Below it, and indented, we'll include a selector node so it only chooses to display one string.

```
19 - World:  
20 -     selector:
```

The selector will have an array of nodes to choose from, so each node we make will be indented and the first line will have a hyphen before it. Our nodes will have three parts to them: a condition, an effect text, and effects. The condition will be the boolean values we made earlier, the effect text will be a string using your name, and the effects will be setting the conditional boolean to false. You might be wondering why we'd set this variable to false, the answer is that without this code, the variable would always be true once it's been selected once. Additionally, once the node has been taken once, no other node will be taken until a condition can be met, so no changes occur until the user clicks one of the buttons. If you believe you've finished the example and would like to check your code against the completed example, you can find it in the villanelle_standalone\examples folder or by opening the file through the editor.

FAQ / Troubleshooting

- Most common/likely problems along with fixes
 - Section requires more playtesting and user trials to be filled out

Provide Feedback

If you have any comments or concerns about the Villanelle application, feel free to contact us at villanelleteam@gmail.com. We are always open to feedback and strive to maintain an open dialogue with the development community in order to produce the best product possible.

Once you've experimented with the Villanelle application, we would like to encourage you to fill out the following survey: <http://bit.ly/VillanelleSurvey> We will use the results of this

survey to help guide future development and assess the importance of certain features to our users.

Villanelle Development Team

Sasha Azad

Claire Christopher

Maddie Ingling

Owais Iqbal

Rook Liu

Dr. Chris Martens

Tony Mosolf