Text Adventure Creator Documentation

“Text Adventure Creator” is a tool for creating modern text adventures through a flowchart interface. This document will guide you through the process of creating a simple text adventure with the tool to get you started.

# Creating a new game instance

To start your new game project, you’ll first need to create an instance of it within the tool. Open “Text Adventure Creator” and press the “New” button.

You will then be prompted to enter four things:

1. **Game title** – this is a unique title used to identify your game within the tools. While this cannot be changed directly, the title displayed in-game can be edited at any time.
2. **Game developer** – the name displayed in-game as the developer of the text adventure. Ideally your name or your studio’s name. This can be changed at any time.
3. **Invalid input response** – the text shown in-game when the user enters something that your logic isn’t set up to handle. This can be changed at any time.
4. **Disabled input response** – the text shown in-game while the system is typing a response to the user and input is disabled. This can be changed at any time.

Once you’ve entered this information, your new game will be created and selected in the tool’s dropdown.

# Editing game logic

With your game instance selected in the tool’s dropdown, press “Edit Logic”. A new window will open – when it does, double click on “main” in the top left. This is the flowchart of our current game. All new game instances come with a demo flowchart to show you the basic layout of nodes. Click on the “+” next to “Nodes” in the top left to see all available nodes to add to the flowchart.

It’s important to understand the different node types and the logic they provide before editing your game. Read on to find out more.

## Core game structure nodes (game, level, zone, state)

These nodes make up the core of the game, defining information and structure. This is essentially the hierarchy of the game – GAME, LEVEL, ZONE, STATE. Zones are made up of states, levels are made up of zones, the game is made up of levels.

* **Game** – this is the initial starting point of any flowchart. Make sure it is placed first. Game provides two parameters:
  + **Title** – this is what you entered when creating your game instance – the game title.
    - Be aware this will only change the title in-game, and not in the tools.
  + **Developer** – this is what you entered when creating your game instance – the game’s developer.
* **Generic Responses** – this should be linked after the **Game** node, it provides two parameters:
  + **Invalid input** – this is what you entered when creating your game instance – the response for an input that isn’t supported by your logic.
  + **Disabled input** – this is what you entered when creating your game instance – the text to display when the input is disabled.
* **Game Level** – this should also be linked after the **Game** node. A game can have as many levels as you like, they are loaded in sequence so the level at position “2” on the **Game** node will be the default level. This node provides one parameter:
  + **Level name** – the name of this level to use as a reference when moving the player.
* **Play Zone** – this should be linked after a **Game Level** node. A level can have as many zones as you like, they are loaded in sequence, so the zone at position “1” on the **Game Level** node will be the default zone. This provides one parameter:
  + **Zone name** – the name of this zone to use as a reference when moving the player.
* **Play Zone State** – this should be linked after a **Play Zone** node. A zone can have as many states as you like, they are loaded in sequence so the state at position “1” on the **Play Zone** node will be the default state. This provides one parameter:
  + **State name** – the name of the state to use as a reference when moving the player.
* **Zone Introduction Text** – this should be linked after a **Play Zone State** node. This defines the text you see upon entering a zone at a certain state. Note this intro text is placed in each state, not in the zone itself. This provides one parameter:
  + **Intro text** – this is a localised string for the introduction text. Enter the string ID here from the localisation editor – do not enter plain text, it will not be interpreted by the compiler.

## Game user interaction nodes

These nodes all begin the logic branches within a zone state. Both must be defined, even if the subject is not required to trigger the logic. Multiple subjects can be defined for one action to trigger different logic for different subjects.

* **User Input ‘Action’** – this is the foundation of the input logic and should be linked after a **Play Zone State** node. This node defines the “action” text that the user must type to trigger this branch of logic. Provides one parameter:
  + **Input action** – the “action” text from the user’s input - for example, if the user inputs “GO TO HOUSE”, the “action” text here would be “GO TO”. It’s best to keep these “action” triggers uniform throughout your game to make it easily understandable for the player.
    - Examples include: “GO TO”, “LOOK AROUND”, “OPEN”, “USE”, “LOOK AT”
* **User Input ‘Subject’** – this is another core foundation to the input logic and should be linked after a **User Input ‘Action’** node. Be aware you can link multiple “subject” nodes to an “action” node, this stops you needing multiple “action” nodes defining the same string. This node defines the “subject” text of a user’s input to trigger this branch of logic. Provides one parameter:
  + **Input Subject** – the “subject” text from the user’s input – for example, if the user inputs “GO TO HOUSE”, the “subject” text here would be “HOUSE”. Subject can be left blank – but this node still must be placed.

## Game logic and response nodes

These nodes all define the logic to perform once the input conditions have been met through the “action” and “subject” nodes.

* **Text Output Response** – this outputs text to the user in the game’s response area and should be linked after a **User Input ‘Subject’** node. Provides one parameter:
  + **System response** – this is a localised string to output to the player. Enter the string ID here from the localisation editor – do not enter plain text, it will not be interpreted by the compiler.
* **Does Game Data Exist** – this is a decision node with a true/false output. It runs logic based on the user’s “game data” and allows for more depth to the game logic. It provides two parameters:
  + **Data requirement** – this determines if the “game data” identifier specified on this node is OPTIONAL or REQUIRED. If OPTIONAL, any other logic on the input’s logic branch will run as well as this node’s output no matter if the user has the specified “game data” or not. If REQUIRED, the output of this node will run, but any other logic will be disabled on the branch if the user doesn’t have the specified “game data”.
  + **Game data** – this is the identifier for the “game data” which is either REQUIRED or OPTIONAL. This identifier is a string and should be consistent with any other mentions of this “game data” in other nodes.
* Game Data Action -
* Move To
* Referenced Action