QUEST SYSTEM

Definition for the quest's json. The quest system will receive some initial variables for setup and will start at the first node in the array. The navigation between nodes will be by nodeId.

File location

All quest files are on the path below:
<GAME DIRECTORY>\Endless2\Dialogues\

The quests are found by the name they have as the key, so no 2 quests should have the same archive name. Only ".quest" archives are considered, any other files here will be skipped.

If any quest error is detected it will be contained on the path below:
C:\Users\<YOUR USER>\AppData\Local\ENDLESS_2\Endless2\QUEST ERRORS.txt

It's strongly advised to use a Json Editor to check for errors or bad json formatting.

Node base

Base variables for each node.

- nodeId: the id of the current node.
- nodeType: signal the type of object the node is.
- nodeNextChained(optional): does an immediate call to this node.
- questDescAux(optional): changes the text from the Quest Watch Box in the map. It's good/advised to use it for step-by-step instructions for the player.

Node: setup

It has to be the first node. It will validate if some extra variables are needed and will immediately call the nodeNext.

- category / subcategory: for filtering and categorising the quests.
  Values:
  - test: will be filtered out unless manually added.
  - contract: for Town contracts.
    - tier1: quests from lvl 0-3, Sum 0-12.
    - tier2: quests from lvl 4-6, Sum 13-24.
    - tier3: quests from lvl 7-10, Sum +25.
  - event: for map events.
    - specific: won't be called randomly, just by specific circumstances (like a new game).
    - timed: will feed the Event Engine that prompts timed quests from time to time.
  - quest: for specific quests called by dialogues or code.
- event: necessary when category is 'event' and subcategory is 'timed'
  - cooldown: number of days. Will be set after the event is called.
  - requirements: an object of key-value pairs that allows the trigger of the event.
    - unique: if "true", verify if the Quest was already done (with Success or Failure), if so, then it can't be triggered.

        - questSuccess: the Quest Id to look for to trigger.
        - questFailure: the Quest Id to look for to trigger.
        - minimumDays: number in days, can only be triggered after these days are greater.
        - isNight: check if it's night. Values "true" or "false".
        - isRaining: check if it's raining. Values "true" or "false".
        - biome: validate the biome.
            - Values(caseless): Water, Plains, Desert, Forest, Mountain.
        - moon: validate the moon.
            - Values(caseless): New, Crescent, Full, Gibbous.
        - memberRace: check if any party members are from a specific race.
            - Values(caseless): Elven, Dwarven, Sylvan, Human, Orc, Dark Elven, Goblin
        - memberAmount: number of live party members.
        - nearTown: if "true", verify if a Town is at 7 squares from the player. Add the town name and race to the Quest variableBag as eventTownName and eventTownRace.
- variables(optional): an array of strings that have all the required variables for the dialogue.
- variableBag(optional): an object with key and value to the quest starts with.
- nodeNext: on setup end, which node will be called.
- nodeFail: on a failure state, which node will be called.
- expireDays: the days before the quest expires.
- questName: the quest name to be displayed.
- questDesc: the quest description to be displayed.
- questDescLong: the quest long description to be displayed.
- outcomes: an object that holds the prizes and penalties.
    - reward: the rewards as an array of outcome objects.
    - penalty: the penalties as an array of outcome objects.
- protoPartyB(optional): an array of ProtoChars. Creates a party B based on this object (see protoParty on Dialogue Engine doc).
- protoPartyC: like protoParty, but creates another unrelated party.

Object: outcomes

A collection of rewards and punishments. Variables, marked as '<>', will be replaced on subtype, amount and stringValue.

- type: a string type for the outcome.
    - COINS: adds or subtracts coins from the party.
    - XP: adds XP to all party members.
    - TOWN_REPUTATION: change the Town reputation to the player.
    - MOOD_INC/MOOD_DECR: increase or decrease mood by 1.
    - ITEM: adds an item to the party.
    - TEXT: a text to be shown as an outcome.
    - QUEST: a quest to be added by Quest ID.
- subtype(optional): a string subtype for the outcome
    - TOWN_REPUTATION: variable string that contains the town name.

    - ITEM: Item ID.
    - QUEST: the quest ID to add to the party.
- amount(optional): a numerical value for the outcome. Can be negative or positive.
    - Obligatory for: COINS, XP, TOWN_REPUTATION
    - MOOD_INC/MOOD_DECR: determine the chance of a mood change.
    - ITEM: items that are not stackable will be set to 1.
- stringValue(optional):
    - ITEM: the item tags separated by ",".
        - Example: "BLACK_TAINT,KEEN,HEAVY".
    - TEXT: a rich text value to be displayed.
    - QUEST: as a variableBag Json.

Node: waypoint

Create a map waypoint that is triggered by proximity.

- nodeNext: which node will be called when the way point is reached.
- distance: as a number of cells from the player. Will avoid being placed at the water, towns evil centers and other waypoints. Will shrink its radius until a place is empty and possible. Alternatively this can be a variable name that is set inside the game or by the Dialogue.

Node: waypointTown

Create a map waypoint that is triggered by proximity. The Quest Engine will try to gather the townName property from the variableBag set inside the game and add a waypoint on top of that Town.

- nodeNext: which node will be called when the way point is reached.
- townName: the variable name, between '<>' that is set inside the game or by the Dialogue.

Node: timer

Create a map timer that is triggered when the time runs out.

- nodeNext: which node will be called when the time is up.
- hours: as a number of hours before it's triggered.

Node: dialogue

When on map, calls a Dialogue and send it the following variables:

- questUniqueId
- the variableBag from party current quest
- all nodeNext from the dialogue node

The variables from this node are as follows:

- dialogueId: the dialogue id that should be called.
- nodeNext(*): any variable that starts with 'nodeNext' will be also sent to the dialogue. These variables can be used as a callback (var. callbackQuestNode) on the Dialogue side.
- actorsA/actorsB(optional): as an array of objects. If present, will set up the parties sent to the Dialogue. If a character is selected, it will not be available to be added again. So the sequence of adding is important.
    - party(mandatory): Where the actor will be from.

        - Values A or B.
    - race(optional): Select a character by race.
        - Values(caseless): Elven, Dwarven, Sylvan, Human, Orc, Dark Elven, Goblin
    - charUniqueId(optional): Select a character by UniqueId.
    - getAll(optional): if present, will add all the remainder characters to the actor group.
- reverseParties(optional): if actorsA/actorsB are present it will flip the parties before calling the Dialogue.

Node: troop

Create a troop on the map that can only be attacked by the player. The following variables will be sent to the troop:

- questUniqueId

The variables from this node are as follows:

- distance: as a number of cells from the player. Will avoid being placed at the water, towns evil centers and other waypoints. Will shrink its radius until a place is empty and possible. Alternatively this can be a variable name that is set inside the game or by the Dialogue.
- party(optional): B (default) or C. The party that will be fought. It's created on the setup node.
- isAnimal(optional): set the sprite of the troop to be an animal.
- nodeNextOnVictory: what quest node to callback on a Victory.
- nodeNextOnFlee(optional): what quest node to callback on a Flee.

Node: listener

A node that will wait for a trigger to be raised by a specific event. After the trigger, it will call the nodeNext.

- nodeNext: what quest node to callback.
- type(caseless): the string that will be listened to
    - Tavern: when the town tavern is called.
        - townRace(optional): the town race to be checked.
            - Values: ELVEN, DWARVEN, SYLVAN, HUMAN, ORC, DARK ELVEN, GOBLIN.
    - Inn: when the in town inn is called.
    - Dialogue: when a dialogue is called.
        - dialogueId: the dialogue ID called.
    - BuySkill: when a Skill is bought in the town.
    - EnterCaves: when the player enters the Infinite Caves.
    - EnterRuins: when the player enters any ruin.
    - DialogueOutcome: when a specific dialogue outcome is called.
        - stringValue: the string value from the outcome.
    - CombatAfter: when a combat ends.
        - objectiveMode: if present, verify the Master objective (ex.: 0005-SKILL MASTERY).

Node: setVariable

This node sets a variable inside the variableBag. If the variable doesn't exist, it creates it.

- variableName: the variable name.
- variableValue: the string variable value.
- nodeNextChained: which node will be called right after this one. Despite it being optional, it's good practice to add this to go to other nodes.

Node: setTroopPlayerVar

Set any variable within objTroopPlayer, the troop representing the player on the map.

The variables from this node are:

- nodeNextChained(optional): which node will be called right after this one. Despite it being optional, it's good practice to add this to go to other nodes.
- variableName: any string.
- variableValue: any value.
    - Booleans have to be set as "true" or "false" due to a GameMaker limitation that interprets those as 1 and 0.

Node: ender

Ends the current quest and adds it to the completed quest pool.

The variables from this node are as follows:

- endType: values are Success or Failure.
    - Success will trigger the "reward" outcome from the setup node.
    - Failure will trigger the "penalty" outcome from the setup node.
- text: the user readable text to be stored on the completed quest. Any variable between "<>" will be replaced by quest variables if they exist.

Town Contracts

When making quests for Town Contracts some variables will be provided by the game.

- townQuest: the town name that was the quest giver.
- townQuestRace: as string, can be used in protoChars to replace values.

The following variables need to be present at the 'variables' array on the setup to be provided.

- townA, townB, townC, townD, townE: Those will be a random 5 closest towns from the current town.